AMILCAR FERNANDES COSTA DE ABREU

INDEXAÇÃO DE IMAGENS DE PROFUNDIDADE BASEADA EM CONTEÚDO

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Informática, Curso de Pós-Graduação em Informática, Setor de Informática, Universidade Federal do Paraná.

Orientador: Prof. Dr. André Luiz P. Guedes

CURITIBA

2003

Ministério da Educação
Universidade Federal do Paraná
Mestrado em Informática

PARECER

Nós, abaixo assinados, membros da Banca Examinadora da defesa de Dissertação de Mestrado em Informática, da aluno Amilcar Fernandes Costa de Abreu, avaliamos o trabalho intitulado, "*Indexação de Imagens de Profundidade Baseada em Conteúdo*", cuja defesa foi realizada no dia 30 de setembro de 2003, às dez e trinta horas, no Auditório da Informática da Universidade Federal do Paraná. Após a avaliação, decidimos pela aprovação do candidato.

Curitiba, 30 de setembro de 2003.

Prof. Dr. André Luiz Pires Guedes
DINF/UFPR - Orientador

Profª. Dra. Taniá Mezzadri Centeno
CEFET/PR – Membro Externo

Prof. Dr. Hélio Pedrini
DINF/UFPR – Membro Interno

Prof. Dr. Marcos Sfair Sunye
DINF/UFPR – Membro Interno

MINISTÉRIO DA EDUCAÇÃO - UNIVERSIDADE FEDERAL DO PARANÁ - Setor de Ciências Exatas - Coordenação de Pós-Graduação

"Dedico a Geysa, minha amada esposa, que durante anos incentiva meu crescimento profissional. Ela foi, sem dúvida, a grande inspiração para a realização de mais esta etapa. A Carol e Gabi, minhas filhas, que aceitaram minha ausência, quase constante, em troca de pequenos momentos de amor e alegria"

AGRADECIMENTOS

Durante a realização deste trabalho, contei com o apoio de inúmeras pessoas; na impossibilidade de nomear a todas, deixo-lhes meu agradecimento sincero. Gostaria de agradecer, em especial, ao Prof. Dr. André Luiz Pires Guedes que aceitou o desafio de acompanhar-me, apesar de termos nos encontrado no meio desta viagem. À Prof. Dra. Olga Regina Pereira Bellon, pela orientação inicial. Aos professores do Departamento de Informática da Universidade Federal do Paraná, pelo muito que contribuíram para a minha formação. Aos colegas Everton e Luciano; que a amizade nascida no percurso do mestrado continue sempre a crescer. Ao amigo Olympio de Menezes Neto pelo incentivo, permitindo que eu me ausentasse do trabalho para dedicar-me ao mestrado. Ao Prof. Júlio Cesar Nitsch, do Unicenp, pelo apoio e estímulo. À Geysa pelo estímulo e apoio constantes no meu esforço de conciliar a vida acadêmica e profissional com a vida em família. A meus pais, pelo amor e incentivo. Aos demais familiares e amigos, de quem e por quem tenho tanto carinho e de cujo convívio subtraí tantas horas para a realização deste trabalho. O amor da família foi imprescindível.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE GRÁFICOS

RESUMO

A maioria dos Sistemas de Recuperação de Imagens Baseada em Conteúdo (SRIC) possui um processo de recuperação em bancos de dados de imagens que pode ser sumarizado em dois passos: Primeiro, a *indexação*: para cada imagem no banco de dados, um vetor é criado, contendo as propriedades essenciais da imagem, e é armazenado em uma estrutura adequada; segundo, a *busca*: dada uma imagem de exemplo, seu vetor de características é computado, comparado com os vetores de características guardados no banco de dados e as imagens similares àquela usada na consulta, são retornadas ao usuário.

Enquanto um grande trabalho de pesquisa é feito em torno da extração automática de características de imagens em profundidade e na elaboração de modelos de representação dos objetos nas imagens, um pequeno esforço é dedicado à combinação destas representações e modelos com estruturas de indexação eficientes.

Este trabalho está inserido no projeto SRIC3D, um sistema de recuperação de imagens tridimensionais desenvolvido pelo Grupo de Processamento de Imagens (IMAGO) da UFPR e tem por objetivo principal prover uma estrutura de armazenamento das representações de imagens 3D e modelos 3D, e de uma estrutura de indexação, capaz de auxiliar o mecanismo de busca do sistema, no atendimento aos diversos tipos de consultas aos quais ele será submetido.

ABSTRACT

Most of the Content-based Images Retrieval Systems of (CBIRS) has a recovery process in image databases that can be summarized in two steps: First, the indexation: for each image in the database, a vector is created, contends the essential properties of the image, and it is stored in an appropriate structure; second, the search: given an image as example, its characteristics vector is computed, compared with the characteristics vectors kept in the database and its similar images are come back to the user.

While a great research work is made around the automatic extraction of characteristics of range images and in the elaboration of models of representation of the objects in the images, a small effort is dedicated to the combination of these representations and models with efficient indexation structures.

This work is inserted in the project SRIC3D, a system of recovery of three-dimensional images developed by the Image Processing Group (IMAGO) of UFPR and it has for main objective to provide a structure of storage of the representations of images 3D and models 3D, and of an indexation structure, capable to aid the search mechanism of the system, in the attendance to the several types of queries to the which it will be submitted.

1. INTRODUÇÃO

Conforme [Zachary & Iyengar, 1999] a Recuperação de Informação (*Information Retrieval*) é "o processo de converter uma requisição de informação em um significativo conjunto de referências". Este conceito pode ser aplicado tanto para a recuperação de informações textuais como também para imagens. O fato de que as "imagens digitalizadas e vídeos, ou seja, objetos visuais, têm se tornado tão importantes quanto às informações textuais tradicionais", gerou a necessidade de desenvolver sistemas capazes de prover o armazenamento, a organização e a recuperação destes objetos.

Um objeto visual pode conter dois tipos de informações: meta-dados e características visuais. Meta-dados, ou atributos textuais a respeito de um objeto, são interpretações sobre a imagem, extraídas com o auxílio humano. Por outro lado, características visuais representam informações implícitas no objeto e são obtidas através de algoritmos automáticos de extração de características. Existem, então, duas técnicas para a recuperação de imagens: baseada em atributos (*atribute-based*) e baseada nas características visuais do objeto (*feature-based*) ou seja, em seu conteúdo (*content-based*).

[Rui et al., 1999] apontam que as pesquisas em Recuperação de Imagens (*Image Retrieval*) surgiram antes mesmo dos anos 70, com o envolvimento das comunidades de pesquisa em Bancos de Dados (*Database Management*) e Visão Computacional (*Computer Vision*).

As primeiras estratégias de recuperação de imagens baseadas em atributos consistiam em fazer manualmente anotações sobre a imagem e armazenar estas anotações em um banco de dados convencional. Utilizando os recursos dos Sistemas Gerenciadores de Bancos de Dados (*Database Management Systems* - DBMS)

convencionais era possível armazenar, indexar e recuperar uma imagem através dos dados anotados nestes campos textuais.

Esta técnica se tornava bastante ineficiente por dois aspectos: primeiro, o esforço despendido para fazer as anotações nas imagens quando o número de imagens a ser armazenada crescia; segundo, a subjetividade humana na interpretação de uma imagem, tanto no momento de inseri-la na base de dados, como também no momento da elaboração de uma consulta.

As estratégias baseadas nas características visuais da imagem, também chamadas de recuperação de imagens baseada em conteúdo (CBIR – *content-based image retrieval*) por outro lado, aplicam algoritmos de processamento de imagens para extrair as informações que serão usadas para caracterizar o conteúdo do objeto e para formar uma alternativa para a sua representação, facilitando o processamento no momento de sua recuperação.

Atualmente, existem muitos Sistemas de Recuperação de Imagens baseada em Conteúdo (SRICs) que utilizam as mais diversas e variadas técnicas para a recuperação de imagens. Existem, entretanto, diferenças entre o tipo e a natureza das informações textuais em relação às informações visuais de uma imagem. Segundo [Zachary et al., 1999], "as informações textuais podem ser consideradas como um vetor unidimensional de palavras ou *token*". Por isso, é consenso da maioria dos pesquisadores, que as imagens e as informações visuais extraídas delas devem ficar armazenadas em bancos de dados adequados ao tipo de informação e ao tipo de consultas às quais será submetido. Neste sentido, observamos que a representação das características visuais de uma imagem pode ser mais complexa, requerendo, em alguns casos, a manipulação de dados espaciais, tais como linhas, círculos, retângulos, polígonos, etc.

Os sistemas de informação baseados em imagens necessitam, então, de mecanismos eficientes de armazenamento, pesquisa e recuperação para estes tipos de

dados, similarmente aos sistemas de informações convencionais – baseados exclusivamente em texto.

Pode-se definir, então, como requisitos fundamentais, necessários para um SRIC: a Extração de Características, os Métodos de Acesso às Informações armazenadas e os Mecanismos de Consultas. Desta forma, as pesquisas em recuperação de imagens estão concentradas basicamente em resolver três questões básicas:

- Qual a melhor representação do conteúdo (informação) de um objeto?
- Quão preciso e eficiente é o processo de recuperação?
- Como pode um usuário especificar um consulta de uma informação visual?

A primeira questão, que será apresentada nos Capítulos 2 e 3, diz respeito às principais características dos SRICs e às informações que podem ser extraídas de uma imagem. Pode-se adiantar que não existe uma "melhor" característica para prover uma representação precisa em um determinado conjunto de imagens. Geralmente uma combinação de características é o mínimo necessário para prover uma representação que proporcione um resultado adequado na recuperação.

O Capítulo 4, diz respeito aos mecanismos que devem ser implementados para agilizar a busca das imagens no banco de dados, pois existem diversos fatores que podem influenciar no processo de recuperação.

Como a eficiência de um SRIC, em última instância, depende da técnica de indexação implementada no banco de dados, muitas pesquisas realizadas nesta área resultaram em uma série de mecanismos, chamados de Métodos de Acesso, que suportam a indexação baseada em conteúdo e respondem bem aos tipos de consultas que podem ser inferidas ao sistema.

Devem existir mecanismos para auxiliar ao usuário na formulação de uma consulta baseada nas mesmas características usadas para representar as imagens. As consultas ao banco de dados para a recuperação de informações possuem algumas

características peculiares e os tipos de consultas existentes serão assunto discutido no Capítulo 5.

Mesmo existindo muitos métodos diferentes para se implementar um sistema de indexação eficiente no tempo e no espaço, em sua grande maioria, eles ainda são muito dependentes da aplicação. Desta forma, o objetivo deste trabalho é avaliar alguns dos Métodos de Acesso existentes e identificar uma arquitetura para o armazenamento, indexação e recuperação de imagens que se seja adequada às necessidades de um sistema de recuperação de imagens de profundidade, o SRIC3D, desenvolvido pelo grupo IMAGO [80] da UFPR, cuja visão geral pode ser observada na Figura 1.

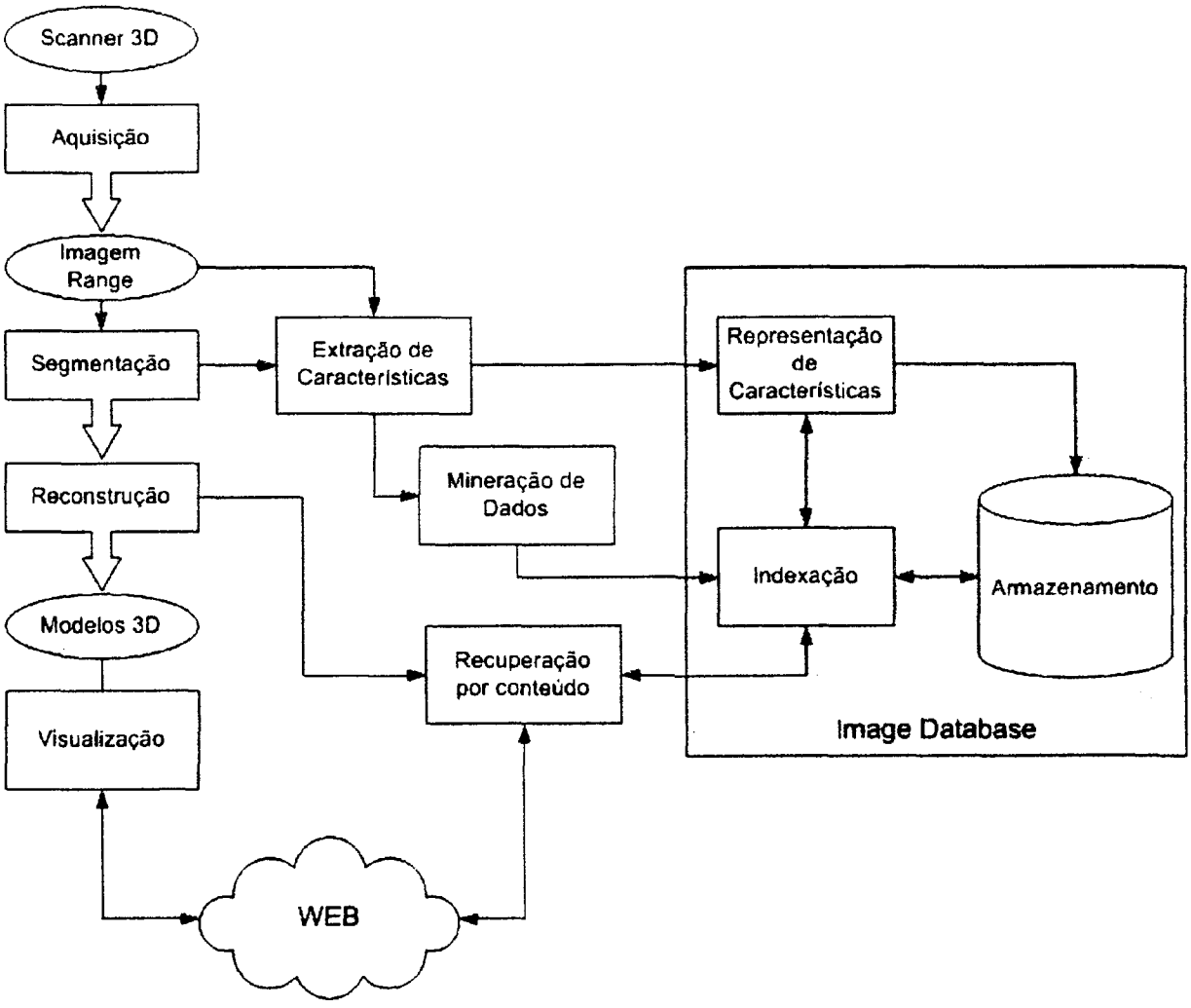


Figura 1: Estrutura do SRIC3D.

O modelo de dados proposto está descrito no Capítulo 6. Os testes realizados sobre a estrutura criada são apresentados nos Capítulo 7 e os resultados no Capítulo 8. As conclusões e sugestões de trabalhos futuros são apresentados no Capítulo 9.

2. SISTEMAS DE RECUPERAÇÃO DE IMAGENS

O conceito de Recuperação de Imagens baseada em Conteúdo (RIBC) só veio a ser desenvolvido no início da década de 1990, [Rui et al., 1999]. De acordo com esta técnica, no momento da inserção da imagem no sistema, são extraídas as principais características visuais da imagem (tais como cor, textura, contorno, forma, curvatura, etc.). Estas características formam o vetor de características (*feature vector*) de uma imagem, que é armazenado junto com ela em uma base de dados adequada. No momento de uma consulta, o vetor de características que representa a imagem de consulta é computado e o SGBD compara este vetor com cada vetor armazenado na base de dados. Esta operação pode ter um custo de processamento elevado se o conjunto de dados a ser pesquisado for grande. A necessidade de otimizar a recuperação dos dados deu origem a diversas pesquisas que resultaram em uma grande variedade de "Estruturas de Indexação" também chamadas "Métodos de Acesso".

Nos SGBDs convencionais, os índices são criados sobre alguns atributos, chamados de chaves de pesquisa (*search keys*), e armazenados juntos com os dados, geralmente, endereçando a localização física dos valores associados a ele. Usando um valor como chave de pesquisa, um registro pode ser rapidamente localizado, [Date, 1984 e Silberschatz, 1997].

No caso dos SRICs, a função da estrutura de indexação também é prover um acesso rápido aos objetos de dados, tornando estes sistemas escaláveis para grandes coleções de imagens. Cada método de acesso procura montar um mecanismo que represente de forma simplificada as informações que endereça e define uma série de operadores relacionais para desempenhar operações de busca na estrutura criada.

Porém, conforme afirma [Petrakis, 2002], "*A recuperação de imagens não é um processo exato (imagens raramente são idênticas)*", desta forma, o mecanismo de busca de um SRIC não pode ser muito rigoroso a ponto de não recuperar *boas-candidatas*, nem tampouco ser flexível demais que recupere muitas *falsas-candidatas*.

O ideal é que seja adaptável ao domínio da aplicação. Ou seja, a estrutura de indexação deve prover um mecanismo de filtragem que elimine de forma rápida as "*falsas candidatas*", reduzindo assim o espaço de dados a ser pesquisado (área hachurada da Figura 2).

Neste caso, deve existir um conceito de *similaridade* implementado no banco de dados do SRIC: uma imagem é similar à outra se os valores de determinadas características são próximos. Uma recuperação eficiente retornaria como resposta à uma consulta todas as imagens cujo vetor de característica se assemelhe ao vetor de característica da imagem de consulta (área cinza escuro da Figura 2).

Resumidamente, a idéia geral de um SRIC é criar para cada imagem um vetor de características que seja uma representação, o mais próximo possível, da imagem armazenada e implementar uma estrutura de indexação e operadores que permitam realizar pesquisas eficientes sobre estes vetores.

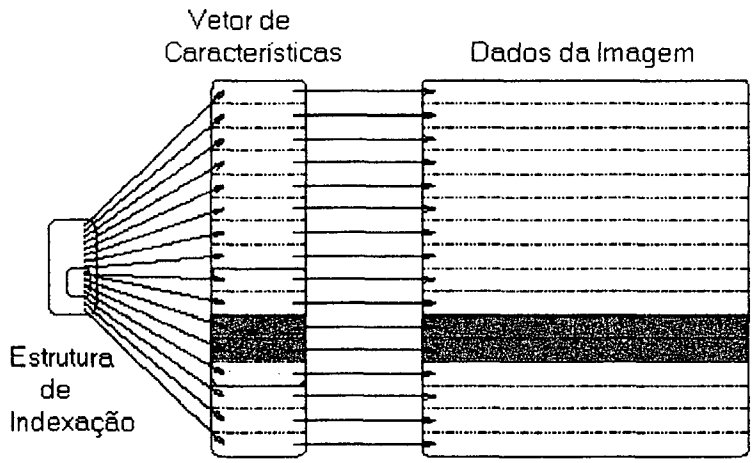


Figura 2 : Estrutura de Indexação e Representação de uma imagem.

A representação de uma imagem é feita por meio de um conjunto de características extraídas por um processo automático composto por um grupo de algoritmos. E este mesmo processo é aplicado sobre uma certa quantidade de imagens, representando estas imagens por vetores de características de mesmas dimensões. Uma vez armazenados na base de dados, estes vetores formam um espaço multidimensional,

onde são realizadas operações de comparação entre os dados. As imagens, então, são tratadas como objetos dentro deste espaço multidimensional.

Um objeto multidimensional pode ser representado por um único ponto no espaço multidimensional ou por uma centena de polígonos, arbitrariamente distribuídos pelo espaço.

Para ilustrar o conceito de espaço multidimensional, temos na Figura 3, um vetor de três dimensões, logo um vetor de características tridimensional, onde os objetos são representados por pontos. O vetor do objeto em questão contém o valor 3.5 para a dimensão X, 0 na dimensão Y e 8 na dimensão Z.

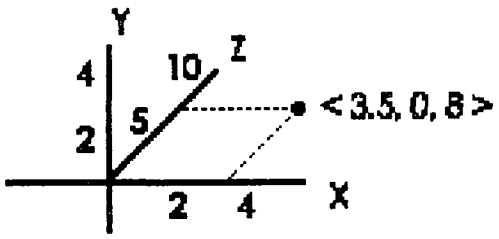


Figura 3: Representação de um espaço Multidimensional

De forma contrária aos dados convencionais, a recuperação de dados espaciais não é usualmente baseada apenas nos valores de certos atributos alfanuméricos, mas sim na localização espacial de um objeto no espaço multidimensional. A recuperação de dados em um banco de dados espacial (*spatial database*) geralmente requer a rápida execução de operações geométricas de busca, tais como consulta por ponto ou por região no espaço multidimensional. Os métodos de acesso multidimensionais serão vistos no Capítulo 4.

Embora o conjunto de características, extraídas por um SRIC, possa ser grande, o conjunto de características capaz de representar uma imagem é pequeno. Esse conjunto, chamado de *dimensão embutida* [White & Jain, 1996b; Rui et al., 1999], é obtido através da transformação do vetor de características em um espaço

dimensional menor, por meio de um novo sistema de eixos, no qual a maior variância dos dados é preservada em poucas dimensões.

Existem na literatura várias técnicas para a redução da dimensão [Faloutsos & Lin, 1995; White & Jain, 1996a; Chandrasekaran et al., 1997], porém as técnicas mais populares são a Transformada KL ("*Karhunem-Loeve Transform*"), o agrupamento inteligente por coluna (*column-wise clustering*) definido em [Dagenhart, 1996] e algoritmos de Mineração de Dados (*Data Mining*), como por exemplo em [Faloutsos & Lin, 1995].

[Aggarwal, 2001] faz uma análise interessante a respeito dos efeitos da redução da dimensão em sistemas de alta dimensionalidade. Um deles é o elevado custo computacional identificado nas possíveis atualizações a que estas estruturas estão sujeitas. Ou seja, na inclusão de novo valor ou conjunto de dados, pode ser necessário recalcular as dimensões (características) a serem utilizadas na representação das imagens.

O desempenho de um sistema de recuperação de imagens pode ser avaliado segundo os seguintes critérios:

- O espaço necessário para armazenar o vetor de característica de todas as imagens;
- O tempo necessário para computar o vetor de características da imagem de consulta;
- O tempo gasto procurando imagens similares à imagem de consulta;
- A relação entre a quantidade de falsas-candidatas e o total de imagens recuperadas;
- O número de imagens similares não retornadas versus o número total de imagens recuperadas;
- O número total de imagens recuperadas versus o número total de imagens examinadas;

Atualmente, existem vários SRICs que utilizam as mais diversas e variadas técnicas para a recuperação de imagens. Para este trabalho, foi preciso avaliar alguns sistemas de recuperação de imagens existentes, procurando observar o comportamento do mecanismo de recuperação utilizado, identificando em cada um, suas vantagens, desvantagens, limitações, tentando identificar uma estrutura que se adequasse à recuperação de imagens de profundidade.

Um sistema bastante conhecido é o QBIC [Flickner et al., 1995], desenvolvido pela IBM. Este sistema provê recuperação de imagens e vídeo baseada numa combinação de cor, textura, forma e um esboço do objeto. Este esboço é feito pelo usuário através de uma interface gráfica. Uma das representações utilizadas é um vetor contendo 20 características de momento, onde é aplicada a Transformada KL, para reduzir o número de dimensões para três, usando como estrutura de indexação uma R*-Tree (detalhes no Capítulo 4).

Outro sistema existente é o Chabot [Ogle & Stonebraker, 1995]. Este sistema armazena informações textuais a respeito das imagens e seu histograma de cores. Através da quantização do histograma de cores são criados conceitos de cores predominantes (*Mostly*) e cores presentes (*Some*) na imagem. Estes conceitos são combinados com as informações textuais para permitir a recuperação das imagens. Com relação a estrutura de indexação, não existe uma menção específica sobre qual é utilizada, mas pelos tipos de dados empregados, os índices *B-Tree* são mais adequados, conforme será discutido no Capítulo 4.

A Universidade da Columbia (USA) desenvolveu dois sistemas, o VisualSeek [Smith & Chang, 1996] e o WebSeek [Smith & Chang, 1997]. Em ambos são utilizadas características globais baseadas no histograma de cores e na relação espacial de regiões formadas pela mesma cor. A estrutura de indexação é provida por uma árvore binária[Date, 1984].

O MIT Media Laboratory também deu sua contribuição desenvolvendo o Photobook [Pentland et al., 1994]. Este sistema divide-se em três módulos:

recuperação por características de forma, recuperação por características de textura e reconhecimento de faces. A consulta inicial é feita quando o usuário escolhe uma imagem de uma das categorias definidas no sistema. Este, por sua vez, retorna um conjunto de imagens pertencentes àquela categoria, baseado nas anotações existentes para cada uma. Após serem apresentadas as *imagens-candidatas*, o usuário pode refinar sua busca escolhendo o conjunto das imagens que mais se aproximam do que ele procura. Nesta segunda etapa, o sistema pode utilizar forma e textura para determinar a similaridade entre as imagens ou pode utilizar algoritmos para o reconhecimento de faces, caso tenha sido este o módulo escolhido. Podendo, ainda, combinar estas funções de similaridade com as informações textuais anotadas em cada categoria.

Em [Swets et al., 1996] é sugerido um sistema de Banco de Dados para imagens que combina a utilização de informações alfanuméricas associadas às anotações feitas junto com a imagem e também, através de características extraídas através de uma imagem exemplo. O método usado neste sistema decompõe automaticamente um vetor de características de alta dimensionalidade em uma estrutura de árvore composta por vetores menores que representam as características mais significantes (MEF - *most expressive features*) e as mais discriminantes (MDF – *most discriminat features*) da imagem através de um método de análise de discriminante hierárquica.

Em [Ahmad & Grosky, 1997] é apresentada uma nova representação da imagem. Nesta técnica, a imagem é recursivamente decomposta em um arranjo espacial de características de pontos enquanto preserva as relações espaciais entre seus vários componentes. Uma árvore do tipo "*Quad-Tree*" [Samet, 1994] é utilizada para gerenciar a decomposição hierárquica e ajuda a quantificar as medições de similaridade.

Em [Alwis & Austin, 1999] são utilizadas redes neurais para a recuperação de informações no banco de dados.

Além dos sistemas apresentados existem outros, conforme [Pereira, 2001] apresenta, porém, nenhum destes tem condições de atender a todas as necessidades do SRCI3D, pois foram desenvolvidos para aplicações específicas, mesmo aqueles que manipulam imagens 3D.

Outra necessidade do SRIC3D é que ele seja flexível e permita que sejam inseridas novas características sem que seja necessário fazer grandes mudanças no sistema e isso não foi encontrado em nenhum deles.

3. EXTRAÇÃO DE CARACTERÍSTICAS

Uma característica de uma imagem é uma propriedade distinta ou um atributo que exprime uma aparência visual da imagem. Algumas características são consideradas naturais, pois representam informações que são percebidas e extraídas diretamente da imagem, tais como: a cor de uma região da imagem ou repetidas diferenças de tons, compondo uma determinada textura, etc. Outras características são consideradas artificiais, pois são resultados de uma manipulação computacional da imagem, como, por exemplo, a quantidade de pontos de mesma cor.

A extração de característica em seu sentido mais formal é a criação de uma representação para os dados originais (imagem), ou uma transformação deles, em um novo tipo de dado. A extração de características constitui, então, um processo importante na montagem de uma base de dados de imagens. Nesse sentido [Smeulders et al., 2000] e [Petrakis et al., 2002b], afirmam que as técnicas de extração de características, ou mesmo as próprias características podem influenciar na eficiência de um SRIC: seja no momento de uma consulta onde o mesmo processo de extração de característica deve ser usado na *imagem-exemplo*; seja na recuperação de uma informação baseada na característica fornecida como parâmetro de pesquisa, para que a representação e o tipo de característica utilizado permitam extrair o maior número de resultados positivos, eliminando também o maior número de falsas candidatas.

Existe uma atenção especial dos pesquisadores nos processos de extração de características e na correta descrição ou representação do conteúdo de uma imagem. Existem, também, diferentes classificações para tais técnicas.

[Zhou et al., 2000] afirmam que existem duas abordagens diferentes: uma trabalha baseada na percepção humana e suas experiências para então escolher a melhor representação para uma característica. E a outra, mais centrada na máquina, onde um esquema computacional é escolhido para a extração de determinada característica. Em outras palavras, esta técnica procura "prover um algoritmo eficiente

capaz de computar os 'números'; e então a efetividade ou a correlação entre os 'números' e a percepção humana são estabelecidos por experimentos" [Zhou et al., 2000].

A área de Visão Computacional classifica as técnicas de extração de características visuais de imagens em dois tipos básicos: primitiva (ou de baixo nível) e lógica (ou de alto nível)[Iqbal & Aggarwal, 2000].

A primeira extrai as informações classificadas como Características de Domínio Geral da imagem, por exemplo: limites, contorno, textura, histograma, cores, etc. A imagem é representada por um conjunto de dados pictoriais não analisados e armazenados em bancos de dados adequados, chamados de bancos de dados pictoriais (*pictorial databases*)[Gaede & Günther, 1998].

Já a segunda, extrai as Características de Domínio Específico, que são abstrações da imagem de acordo com diversos níveis de detalhes. Como por exemplo, a identificação da existência de um objeto do tipo "avião", a representação de uma impressão digital ou uma face. Devemos notar aqui, que existe a necessidade de um conhecimento extra: no caso, a definição de como é um objeto "avião", que arranjo de linhas forma uma face ou indicam uma impressão digital. Estes conceitos são chamados de agrupamento perceptual e se referem à capacidade humana de extrair relações significativas das características de baixo nível da imagem, sem conhecimento semântico sobre ela. Geralmente, estes agrupamentos são realizados por critérios de proximidade, similaridade, continuidade, fechamento e simetria, formando um conjunto de relações significativas [Iqbal & Aggarwal, 2000].

Porém, "por causa da subjetividade de percepção, não existe uma maneira única de representar uma determinada característica de uma imagem. Ou seja, para uma mesma característica existem múltiplas representações as quais podem caracterizá-la sob perspectivas diferentes" [Rui et al., 1999]. Um exemplo disso é a característica cor, que pode ser representada por um histograma ou por um conjunto, ou por momentos, conforme veremos a seguir.

Além disso, geralmente uma única característica não basta para identificar e garantir a unicidade de uma imagem em um banco de dados de imagens (IDB – *image database*).

Desta forma, para inserir uma imagem em um SRIC, devemos armazenar a imagem física (*raw image*) e tantos atributos quantos forem necessários para prover uma recuperação eficiente destas imagens. O conceito de eficiência de um SRIC está relacionado ao fato de que ele deve computar, rapidamente, um conjunto de características que represente a imagem inserida e no momento de uma consulta, deve existir uma função que, aplicada sobre dois conjuntos de características de duas imagens distintas, resulte em um valor tal que seja grande o suficiente se as imagens representadas pelos vetores não são similares e um valor pequeno se elas forem similares.

Boa parte dos SRIC de propósito geral utiliza um conjunto de características primitivas para representar as imagens, com isso, torna-se necessária uma revisão sobre as principais características primitivas e suas representações.

3.1 – CARACTERÍSTICA: COR

A característica cor é usada nos SRICs onde variações de tamanho e orientação da imagem não devam ser consideradas. Existem várias representações ou "arranjos" combinados com métodos para calcular e identificar a similaridade utilizando a característica cor.

3.1.1 – Representação RGB

Neste modelo cada *pixel* da imagem carrega sua informação de cor, que é representada pela intensidade de seus componentes espectrais primários R=*red*, G=*green*, B=*blue*, [Gonzalez & Woods, 1993]. Cada canal de cor pode representar 256 tons diferentes e combinados, é possível representar mais de 16 milhões de cores.

3.1.2 – Histograma de Cores

Um histograma de cores consiste na descrição das cores presentes em uma imagem e a quantidade de *pixels* em cada cor, [Zachary & Iyengar, 1999]. É claro que, a representação de todas as cores existentes em uma imagem é inviável, então existe uma redução no número de tons que podem ser representados em cada canal.

Uma das técnicas mais comuns considera os k *bits* mais significativos de cada canal ($k \leq 8$). Com isso existirão 2^{3k} cores distintas. Por exemplo, se k=3, serão utilizadas 512 cores diferentes. Com isso, um histograma pode ser apenas representado por um vetor com 512 *bins*, onde cada entrada do vetor contém o número de *pixels* de uma determinada cor. Maiores detalhes sobre histogramas de cores podem ser verificados em [Gonzalez & Woods, 1993]e em [Striker & Orengo, 1995].

3.1.3 – Momentos de Cor

Esta técnica parte do principio de que a distribuição de cores em uma imagem pode ser tratada como uma distribuição probabilística, e então caracterizada pelos seus momentos. São extraídos os momentos centrais de maior ordem: o primeiro momento é a média, é usado então a cor média da imagem (*average color*). O segundo momento é o desvio padrão e o terceiro é a raiz cúbica de *skewness* de cada canal. Para a medição de similaridade entre estes conjuntos de dados é usada a Distância Euclideana, [Striker & Orengo,1995].

3.1.4 – Conjunto de Cores

Inicialmente o espaço (RGB) é transformado em um espaço perceptual uniforme e então quantificado, transformando o espaço de cores em M[1] *bins*. O vetor do Conjunto de Cores é binário, identificando a existência ou não de determinada cor na imagem. Complementando a técnica, para a indexação e recuperação, uma árvore binária é criada para prover uma busca rápida [Smith & Chang, 1995].

3.1.5 – Correlogramas de Cores

Em [Huang et al., 1997] foi proposta uma nova estrutura para armazenar as informações de Cor de uma imagem e a relação espacial entre as cores. Basicamente o sistema proposto grava a quantidade de *pixels* de uma cor *j* que está a uma distância *k* de um ponto da cor *i*. Nas palavras do autor, "um Correlograma de cor de uma imagem é uma tabela indexada por pares de cores, onde a *k-ésima* entrada para o par [*i,j*] especifica a probabilidade de encontrar um *pixel* na cor *j* a uma distância *k* de um *pixel* da cor *i* na imagem".

As vantagens neste método se devem ao fato de incluir as relações espaciais das cores e pode ser usado para descrever a distribuição espacial das cores na imagem.

3.2 – CARACTERÍSTICA: TEXTURA

Segundo [Zachary & Iyengar, 1999], a característica textura é difícil de definir, mas muito fácil de percebê-la quando a vemos. A textura se refere a um padrão visual que tem propriedades de homogeneidade que não resulta da presença de uma única cor ou intensidade. Segundo [Jain, 1989], a textura é observada em um padrão estrutural da superfície de objetos tais como madeira, grãos, areia, vidro e

[1] Onde M significa a quantidade de cores analisadas.

nuvens. [Zhou et al., 2000] afirmam que a textura captura a distribuição espacial das variações da luminância na forma especial de "padrões repetidos".

As seis propriedades visuais da textura são: grossura (*coarseness*), contraste, direção, semelhança de linhas, regularidade, e aspereza (*roughness*). A utilização da textura na recuperação de imagens em SRIC tem sido estudada por um bom número de pesquisadores, [Wu et al., 1999]. As técnicas criadas a partir dessas pesquisas podem ser classificadas em quatro tipos, conforme mostra a Figura 4: Métodos Estatísticos, Métodos Geométricos, Métodos Baseados em Modelos e Métodos de Processamento de Sinais[Tuceryan & Jain, 1998].

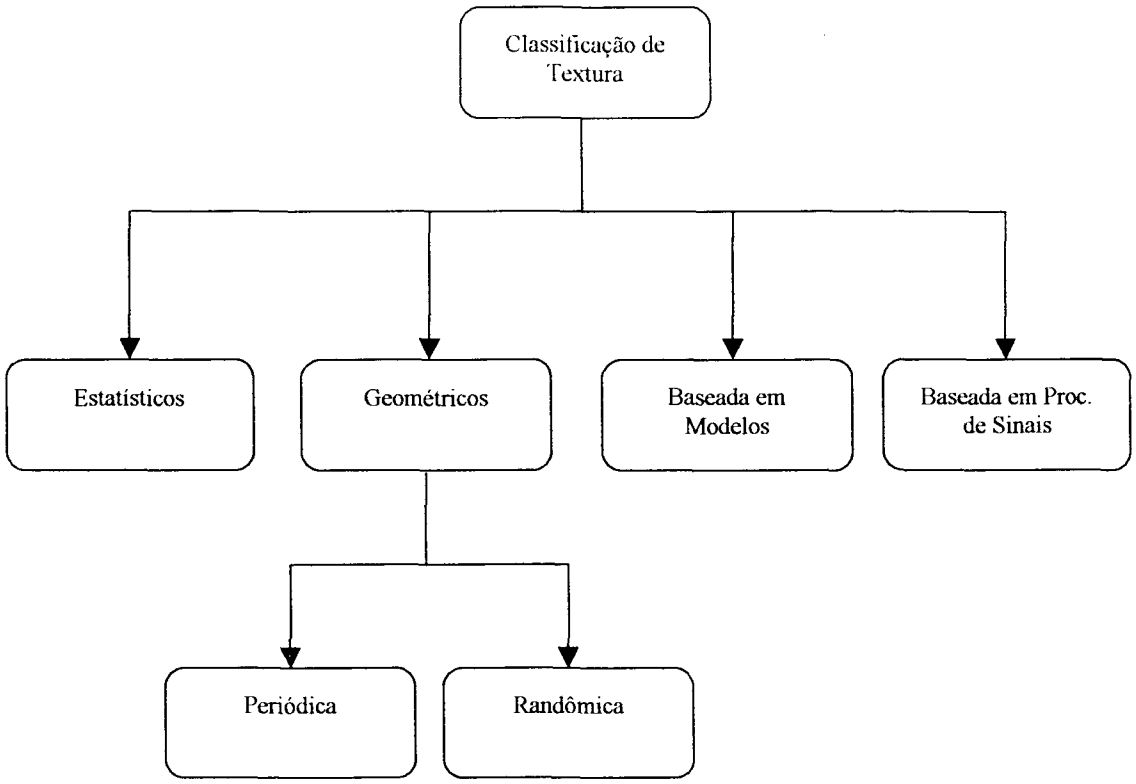


Figura 4: Taxonomia dos métodos de representação de Textura.

3.2.1 – Métodos Estatísticos:

Caracterizam uma textura pela geração da distribuição estatística dos valores de intensidade, assim como a posição e orientação de pontos com valores similares. Nesta categoria estão os métodos: Matriz de Co-ocorrência, Função de

Autocorrelação, Espectro de Energia (*Power Spectrum*), Diferenças Estatísticas (*Difference Statistics*), modelos de Auto-regressão e Transformadas de Fourier.

3.2.1.1 – Matriz de Co-ocorrência:

Uma matriz de co-ocorrência é a totalização dos pontos que possuem um determinado nível de cinza e estão a uma certa distância/direção um do outro. Ou seja, é a relação espacial de N níveis de cinza, obtida estatisticamente e define uma representação da imagem baseada nas estatísticas de segunda-ordem, [Castleman, 1996].

3.2.1.2 – Características de Auto-correlação (*Autocorrelation Features*)

Uma propriedade importante de muitas texturas é a natureza repetitiva da distribuição dos elementos da textura na imagem. A função de Auto-correlação de uma imagem pode ser usada para avaliar a regularidade como também a grossura (*fineness /coarseness*) da textura na imagem, [Wu et al., 1999].

3.2.2 – Métodos Geométricos

Esta classe se caracteriza pela definição de textura como sendo composta de "elementos de textura" ou *primitivas*. O método de análise normalmente depende das propriedades geométricas destes elementos de textura. Uma vez que os elementos de textura sendo identificados na imagem, há duas abordagens principais para analisá-la. Em uma abordagem são computadas as propriedades estatísticas dos elementos de textura e utilizados para caracterizar textura. A outra tenta extrair um padrão de distribuição que descreva a textura, o que pode envolver a aplicação de métodos geométricos ou métodos sintáticos para analisar textura, [Castleman, 1996].

3.2.3 – Métodos Baseados em Modelos

Estes métodos baseiam-se na construção de um modelo de imagem que pode ser usado não só para descrever textura, mas também sintetizá-la. Os parâmetros do modelo capturam as qualidades essenciais da textura.

Nesta categoria estão métodos como: Modelos de Campos Aleatórios (*Random Field Models*) e Fractais.

3.2.4 – Métodos de Processamento de sinal

Segundo [Wu et al., 1999], pesquisas na área de Psico-Física mostraram que o cérebro humano faz uma analise da freqüência da imagem e que a textura adapta-se bem a este tipo de análise por causa de suas propriedades. A maioria das técnicas tenta computar certas características de imagens filtradas, que são então usadas na classificação ou em tarefas de segmentação. Nesta categoria podemos citar: Filtros de Domínio Espacial, Filtragem de Domínio de Fourier, Modelos de Gabor e *Wavelets,* [Jain, 1989]. Um exemplo da aplicação desta técnica pode ser visto em [Wu et al., 1999].

3.3 – CARACTERÍSTICA: FORMA

Forma (*shape*) também é um conceito facilmente entendido, porém é difícil de ser formalizado. Para o ser humano a percepção de forma é um conceito relacionado a informações de alto nível, enquanto que uma definição matemática da forma, tende a descrevê-la com informações de baixo nível. É uma característica específica, que se baseia na "borda" ou "contorno" de um objeto da imagem. Esta definição implica na necessidade de que ocorra um processo de segmentação da imagem para a identificação das bordas ou dos pontos que fazem parte da borda de um objeto na imagem, descrevendo assim a Forma. Em alguns sistemas de Recuperação

de Informações, dependendo da aplicação, é interessante que o contorno do objeto seja preservado ao serem aplicados processos de rotação, translação e escala.

[Brandt, 1999], com base nas avaliações de [Mehtre et al., 1997], classificou as técnicas de descrição de Forma conforme mostra a Figura 5. Segundo o autor, as técnicas podem ser divididas, inicialmente, em *descritores externos*, caso seja descrito o contorno de um objeto, ou *internos* caso seja descrita a área dentro do contorno.

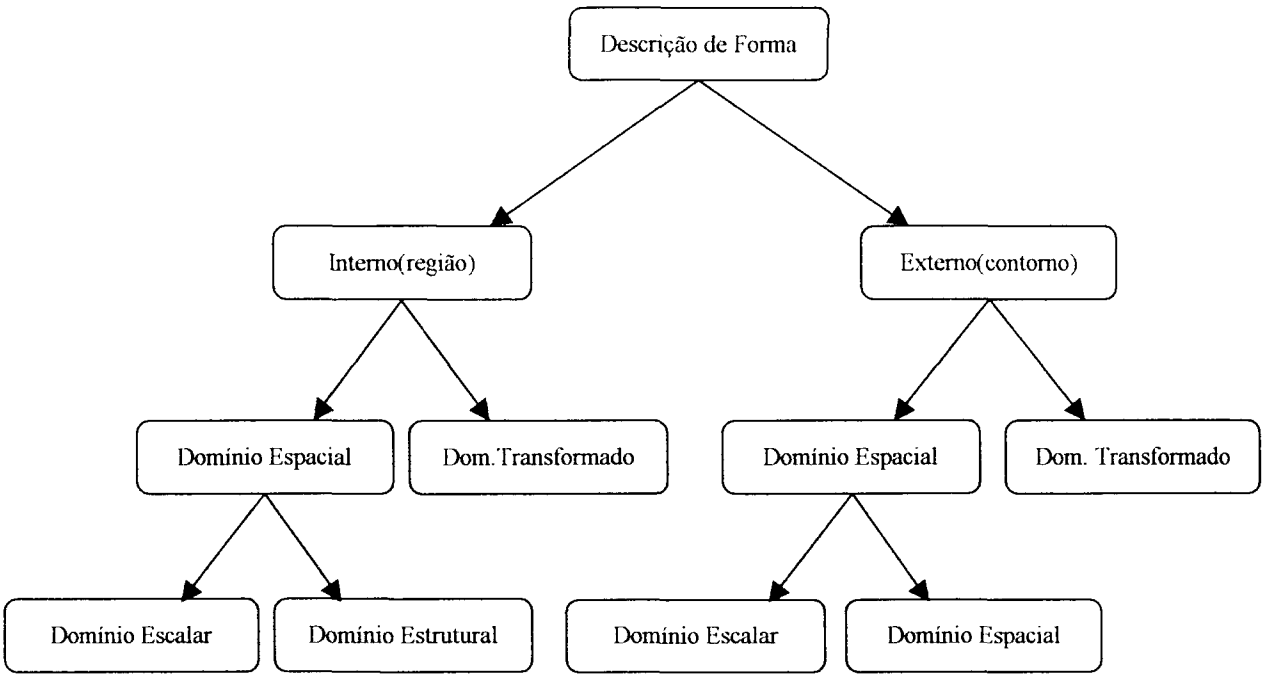


Figura 5: A taxonomia dos métodos de descrição de forma [Brandt.1999].

Ambos descritores ainda são divididos em domínio espacial e domínio transformado. Este último consiste de técnicas baseadas em transformadas matemáticas, tais como Transformada de Fourier [Jain, 1989, Gonzalez & Woods, 1993] e *Wavelet* [Castleman,1996].

As técnicas de domínio espacial interno ou os métodos espaciais baseados nas regiões são usados quando a descrição da forma é feita através da área interna de um objeto. Esta classe pode ser dividida ainda em Estrutural e Escalar. A descrição de

forma, onde toda a imagem é representada por um descritor, pertence também à classe de domínio espacial interno. Esta técnica define a aparência da imagem.

As técnicas de domínio espacial externo são também divididas em métodos de domínio escalar e domínio espacial. Técnicas de domínio espacial incluem descritores que expressam a estrutura e as relações espaciais da forma.

Uma representação de domínio espacial interno é descrita em [Berretti et al., 1999]. Os autores elaboraram uma representação de forma, inicialmente introduzida por [Mehrotra & Gary, 1993], onde, para cada vértice, uma *característica local*[2] é descrita. Esta representação é feita através do ângulo interno do vértice, da distância entre o vértice adjacente e das coordenadas do vértice. Um exemplo desta decomposição do objeto em várias características locais pode ser visto na Figura 6.

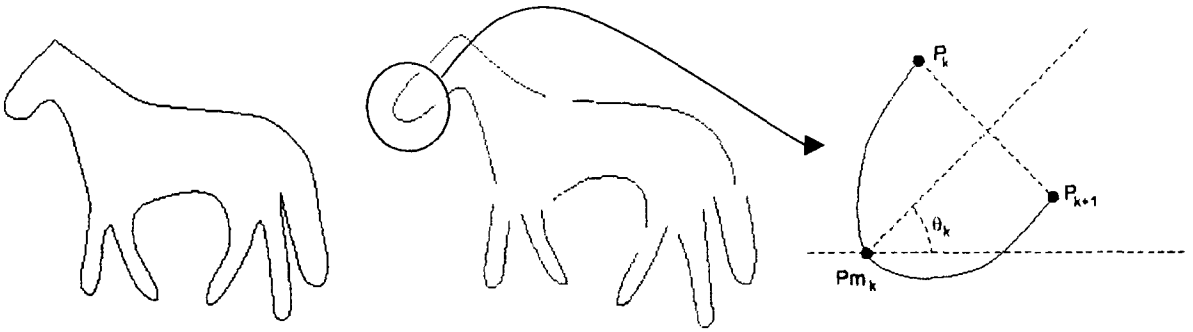


Figura 6: Representação de Forma por [Berretti, et al. ,1999].

Um número fixo destas características locais é extraído de cada forma encontrada na imagem. A forma é então representada por um atributo *string* contendo todas as informações dos cantos ou extremidades escolhidas.

Em [Gonzalez & Woods, 1989] podemos encontrar vários tipos de representação e descritores para forma. Porém, nos sistemas implementados atualmente (ver Capítulo 2), em geral, as características são modeladas através de vetores, que por sua vez, são manipulados como pontos em um espaço multidimensional. A determinação de similaridade entre imagens é, então, feita através

[2] Uma característica local pode ser um canto ou as extremidades do contorno de um objeto.

da redução da dimensão deste vetor de características e depois calculada a distância Euclideana entre estes pontos. Para a recuperação de imagens através da forma, esta combinação é um pouco ineficiente,[Berretti et al., 1999]. Assim, diversas pesquisas procuraram combinar as características de baixo nível, na tentativa de obter uma recuperação de imagens mais eficiente, mas observa-se que a maioria delas apresenta resultados melhores por terem sido adaptadas à aplicação que se destinavam.

Em [Jeong et al., 1999], foram combinadas características de cor, textura, e forma para melhorar a recuperação de imagens. O sistema combina as saídas de três módulos de recuperação reduzindo o número de falsas candidatas. A Figura 7 mostra o diagrama em blocos proposto por [Jeong et al., 1999]. No módulo de COR, são utilizados os momentos (média, variância e *skewness*) das regiões de maior e menor entropia. Estes momentos são comparados com os extraídos da imagem de consulta. O módulo *Wavelet* é usado para recuperar informações de textura. E o módulo *Edge Map* é usado para informação de forma. Os resultados da recuperação de cor, textura e forma, são computados por um método estatístico, que identifica a freqüência com que uma determinada imagem aparece como resultado dos módulos de recuperação, ou seja, se a imagem resulta da recuperação dos três módulos é bem provável que ela seja similar à imagem de consulta.

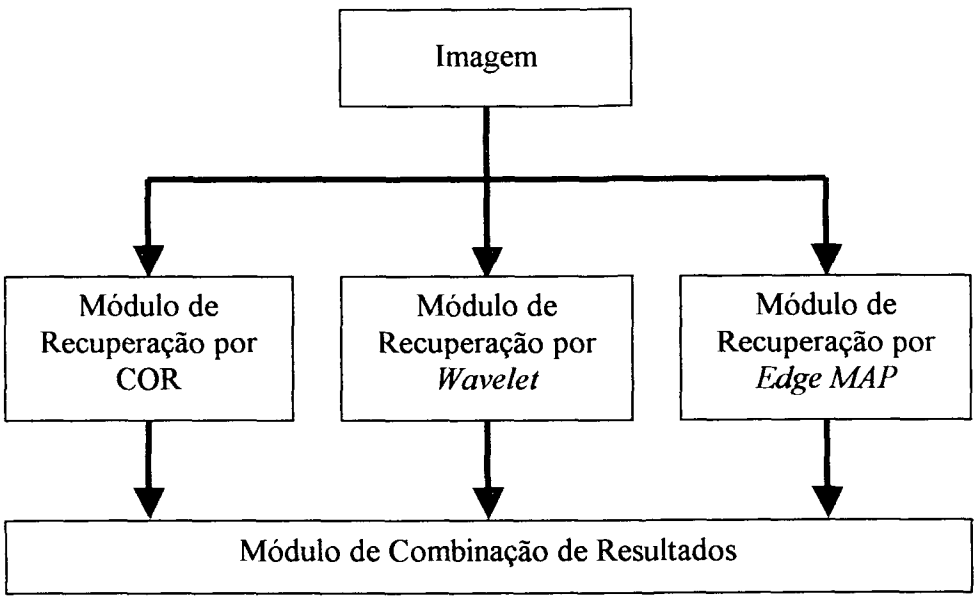


Figura 7: Diagrama em blocos do sistema proposto por [Jeong et al., 1999].

Em [Ma & Zhang., 1998] são feitas comparações entre o desempenho de sistemas que usam cor e textura como características para a recuperação de imagens. Para a característica cor, foram avaliados os métodos de representação: histograma, momentos, vetor de coerência (CCV – *color coherence vectors*) e correlogramas. Este último apresentou melhor desempenho e o vetor de coerência ficou em segundo lugar, com uma estreita vantagem para o histograma.

Para a característica textura foram avaliadas as representações de Tamura (aspereza, direcionalidade e contraste), MRSAR, Histograma Prudente de Extremidades (*Canny Edge Histogram*), Textura de Gabor, Transformada *Wavelet* Pirâmide-Estruturada (PWT) e Transformada Árvore-Estruturada (TWT). A representação MRSAR e de Gabor apresentaram um desempenho melhor.

Podemos então observar que a maioria dos SRIC utiliza um conjunto restrito de características e suas representações, que dependem fortemente da aplicação. Ou seja, deve existir um modelo do conteúdo da imagem de forma a organizar suas características principais e mais relevantes à técnica de recuperação utilizada.

Como este trabalho está inserido em um projeto que prevê a extração de características em imagens de profundidade e o armazenamento destas informações em um banco de dados, é necessário avaliarmos as principais características que podem ser extraídas destas imagens para que possam ser cadastradas no banco de dados para futura recuperação.

3.4 – IMAGENS DE PROFUNDIDADE (*RANGE IMAGES*)

Imagem de profundidade ou imagens *range*, são tipicamente formadas pela varredura de um sensor unidimensional ou bidimensional linearmente a um objeto ou ao redor dele, gerando uma coleção de números que representa a distância de cada ponto da superfície refletora da cena até o sensor. Segundo [Yu et al., 2001], elas são

consideradas pela maioria das técnicas como um matriz de pontos em um espaço tridimensional, onde cada ponto da imagem carrega um valor indicando a distância dele em relação a um ponto de referência.

Segundo [Castleman, 1996], imagens de profundidade podem ser capturadas através de Estereometria, uma técnica onde uma cena tridimensional é fotografada por duas câmeras em posições ligeiramente diferentes, gerando informações sobre a quantidade de luz refletida pelos objetos da imagem. Outras técnicas, para aquisição de imagens de profundidade, utilizam tecnologia *laser* para capturar a distância entre o sensor e a superfície da cena ou objeto.

Para [Jiang & Bunke, 1994], "as imagens de profundidade são apenas um conjunto de números e não é possível obter informações de alto nível através de um processo de interpretação semântica sem algum tipo de percepção organizacional". Essa percepção organizacional pode ser obtida, então, por meio da segmentação das superfícies que compõem a imagem, extraindo informações mais relevantes sobre a imagem.

Existem muitos trabalhos sobre segmentação de imagens de profundidade na área de visão computacional, conforme pode ser visto em [Bab-Hadiashar & Suter, 2000]. Por causa das similaridades com as imagens 2D, técnicas comuns de segmentação de imagens, tais como detecção de borda, regiões crescentes, e agrupamento de pontos podem ser aplicados também para a segmentação de imagens de profundidade [Yu et. Al, 2001].

O objetivo principal das técnicas utilizadas é identificar uma região no objeto que tenha as mesmas características. Os algoritmos de segmentação geralmente são classificados em Algoritmos de Propósitos Gerais e Dedicados. Nas aplicações gerais, apenas o conhecimento geral sobre a superfície é usado para auxiliar a completa segmentação. Por outro lado, aplicações dedicadas procuram por uma estrutura particular nos dados da imagem, tais como planos, cilindros, cones ou sólidos de revolução [Jiang & Bunke, 1994].

3.4.1 – Curvatura

Uma característica importante que é utilizada para a descrição e segmentação de imagens de profundidade é a curvatura de uma superfície. A curvatura não se altera com a aplicação de translação ou rotação da imagem, tornando-se bastante aplicada no SRIC3D, uma vez que o sistema utiliza várias vistas do mesmo objeto. Pesquisas como [Bellon et al., 1999] e [Bellon et al., 2001] demonstram a aplicabilidade dessa característica na segmentação de imagens *range*.

A curvatura de uma superfície geralmente é definida pela Curvatura Gaussiana e pela Curvatura Média (*Mean Curvature*). A curvatura de um determinado ponto varia de acordo com a direção em que ela foi medida. A cada ponto, existe uma direção da máxima curvatura normal identificada por *k1* e uma direção ortogonal, *k2*. k1 e k2 são chamadas então de curvaturas essenciais. A curvatura Gaussiana K e a curvatura Média H são definidas, respectivamente, pelo produto e pela média das curvaturas essenciais.

Através da identificação da curvatura de cada ponto, é possível definir regiões na imagem de mesma curvatura e classifica-las em função dos valores das Curvaturas Gaussiana e Média, conforme a Tabela 1.

H	K	Tipo de superfície
H < 0	K < 0	Sela em cume
H < 0	K = 0	Cume
H < 0	K > 0	
H = 0	K < 0	(Minimum surface)
H = 0	K = 0	(Flat)
H > 0	K < 0	(Saddley Valley)
H > 0	K = 0	(Valley Surface)
H > 0	K > 0	(Pit)

Tabela 1: Classificação de superfícies baseadas em H e K.

3.4.2 – Nuvem de Pontos (*point clouds*)

Esta técnica, interpreta as imagens de profundidade como uma *nuvem de pontos* (*point clouds*) obtida através de um *laser* identificador de distância, onde cada

ponto possui uma coordenada 3D, uma orientação normal calculada da superfície subjacente àquele ponto e o valor da intensidade do *laser* retornado. Esta nuvem de pontos é gerada através da distribuição de pontos com um espaçamento regular e depois estes pontos são conectados a seus vizinhos mais próximos formando uma rede (*mesh*) de triângulos planares,[Yu et al., 2001].

3.5 – EXEMPLOS DE SISTEMAS DE IMAGENS EM PROFUNDIDADE

A maioria dos sistemas que manipulam imagens *range* tem por objetivo a geração de modelos 3D baseada nas vistas de um objeto ou ambiente. Existem vários trabalhos que apresentam diferentes técnicas para recuperar as informações sobre a forma dos objetos nas imagens. Uma destas técnicas, apresentada por [Ramamoorthi & Arvo, 1999] utiliza geradores de modelos (*Generative Models*) desenvolvido por [Snyder, 1992]. Um gerador de modelo utiliza a representação de um objeto através de uma estrutura hierárquica e a generalização das superfícies do objeto, onde as curvas de geração podem ser transformadas continuamente por uma ou várias curvas arbitrárias, gerando uma representação do objeto ou outro objeto similar.

Em [Stamos & Allen, 2000] são capturadas imagens *range* e imagens 2D da cena a ser analisada. As imagens *range* são segmentadas em regiões planares e das imagens 2D são extraídas características de cor de cada região identificada. A combinação destas informações permite a criação de um modelo 3D da cena observada.

Além destes, [Curless & Levoy, 1996] também apresentaram um trabalho para a criação de modelos 3D a partir da distância calculada entre cada ponto e o sensor. Uma outra técnica para a geração de modelos 3D é apresentada em [Zhang & Chen, 2001], onde é utilizado o aprendizado ativo (*active learning ou learning with queries*) através de realimentação de relevância (*relevance feedback*) e a utilização de atributos lógicos (*boolean*), onde algumas anotações são armazenadas caso uma determinada características exista ou não naquele objeto.

Como o objetivo geral destes sistemas não está relacionado com o armazenamento de grande volume de imagens, não existe nenhuma indicação de que exista uma preocupação com o desempenho do sistema na recuperação das imagens.

3.6 – CONCLUSÃO SOBRE EXTRAÇÃO DE CARACTERÍSTICAS

Existem muitas pesquisas para a classificação e extração de um conjunto de características que seja compacto e represente uma imagem eficientemente. Entretanto, estas técnicas não poderiam ser utilizadas em grandes coleções de objetos se não existissem também métodos de acesso adequados aos tipos de dados utilizados e às consultas solicitadas.

4. BANCOS DE DADOS PARA SRIC

Existem diferenças entre o tipo e a natureza das informações textuais em relação às informações visuais de uma imagem. Segundo [Zachary et al., 1999], "as informações textuais podem ser consideradas como um vetor unidimensional de palavras ou um *token*" e o modelo utilizado atualmente pelos SGBDs convencionais para o armazenamento deste tipo de informação é o relacional. Por outro lado, as características visuais são mais complexas. E por causa desta complexidade, um sistema de convencional relacional não é satisfatório para armazenar as características de um conjunto de imagens [Stonebraker & Rowe, 1986].

Quando utilizamos um vetor de características para representar uma imagem, ela se torna um objeto multidimensional em um espaço k-dimensional, onde k é o número de características em cada vetor. Um objeto multidimensional possui uma estrutura complexa. Ele pode ser representado por um único ponto no espaço multidimensional ou por uma centena de linhas, círculos, retângulos ou polígonos, arbitrariamente distribuídos pelo espaço.

Por causa da complexidade existente nos objetos multidimensionais, não é possível armazenar uma coleção destes objetos em uma simples tabela relacional com um tamanho fixo para cada tupla[3]. Mesmo sendo possível simular estes tipos de dados com os tipos de dados embutidos, a elaboração das consultas seria trabalhosa e o desempenho seria prejudicado.

Para ilustrar melhor, deve-se considerar o seguinte exemplo:

O objetivo é armazenar uma coleção de objetos geométricos (por exemplo, polígonos, linhas, e círculos) em um banco de dados convencional relacional. Para isso, seria criada uma relação para cada tipo de objeto com campos apropriados:

---

[3] Em bancos de dados, este conceito significa um registro ou uma linha em uma tabela[Date, 1984].

```
POLÍGONO (id, outros campos);
CÍRCULO  (id, outros campos);
LINHA    (id, outros campos);
```

Exibir estes objetos na tela requer informação adicional que represente cada característica de exibição de objeto (por exemplo, cor, posição, escala, etc.). Como este tipo de informação é a mesma para todos os objetos, pode ser armazenada em uma única relação: EXIBIÇÃO (*cor, posição, escala, tipo_objeto, id_objeto*).

O campo *id_objeto* juntamente com o campo *tipo_objeto* (no caso, "POLÍGONO", "CÍRCULO" ou "LINHA") são os identificadores de uma *tupla*. Dada esta representação, o comando seguinte precisaria ser executado para produzir uma EXIBIÇÃO:

```
foreach OBJ em {POLÍGONO, CIRCULO, LINHA} do
range of  O is OBJ
range of  D is EXIBIÇÃO
retrieve (D.all, O.all)
where  D.id_object = O.id
and  D.obj-tipo = OBJ
```

Infelizmente, esta coleção de comandos não será executada rapidamente por qualquer SGBD relacional para "pintar na tela" uma imagem em real tempo (i.e., um ou dois segundos). O problema é que não importa o quão rápido é o DBMS, mas sim que existem muitas consultas a serem executadas para buscar os dados do objeto. A funcionalidade necessária é a habilidade para armazenar o objeto em um campo na tabela EXIBIÇÃO de forma que só uma consulta seja executada.

Os sistemas de informação baseados em imagens necessitam, então, de mecanismos eficientes de armazenamento, pesquisa e recuperação para estes tipos de dados, similarmente aos sistemas de informações convencionais – baseados exclusivamente em texto.

Outra propriedade é que os bancos de dados espaciais tendem a ser grandes. Pois, em alguns casos, armazenam além da representação da imagem composta por centenas de características, a própria imagem (*raw image*) dentro do banco de dados.

Para se ter uma idéia, uma imagem de 400x400 pontos, onde, para cada ponto, existem 8 bits para representar 256 tons de cinza, teremos uma imagem com aproximadamente 1,2 MBytes. Mesmo utilizando-se imagens em formato compactado (por exemplo: JPEG), um banco de dados com 10.000 imagens superaria a casa dos gigabytes facilmente. No sistema CHABOT [Ogle & Stonebraker, 1995], apresentado no Capítulo 2, cada imagem possui tamanho entre 4 e 6 Mbytes.

No que se refere às operações realizadas nestes bancos de dados, também não existe ainda uma álgebra padrão definida para realizar operações com dados espaciais. "Existem várias propostas feitas no passado, mas ainda não existe um conjunto padronizado de operadores básicos", [Gaede & Gunter, 1998]. O conjunto de operadores ainda em geral é desenvolvido para operar com estruturas de dados que dependem fortemente do domínio da aplicação, embora alguns operadores (tais como a interseção) são mais comuns do que outros. Ainda assim, muitos operadores existentes não são precisos. Segundo [Gaede & Gunther, 1998], a interseção de dois polígonos pode retornar qualquer número de pontos isolados, extremidades não contínuas, ou polígonos disjuntos.

Com relação aos custos computacionais, operadores de bancos de dados espaciais são geralmente mais custosos do que operadores relacionais padrões. Recuperação e atualização de dados espaciais são usualmente baseadas não apenas nos valores de certos atributos alfanuméricos, mas sim na localização espacial de um objeto. A recuperação de dados em um banco de dados espacial geralmente requer a rápida execução de operações geométricas de busca, tais como consulta por ponto ou por região. Ambas operações requerem acesso rápido a esses objetos de dados.

4.1 – ESTRUTURAS DE INDEXAÇÃO

Segundo [Pereira, 2001], o principal objetivo das pesquisas sobre indexação é construir índices diretamente das características dos objetos presentes na imagem. O desafio dos sistemas de recuperação de imagens é o desenvolvimento de mecanismos

que identifiquem as características mais relevantes que agrupam ou classificam melhor um conjunto de imagens e implementem a criação automática de uma estrutura de indexação.

A expectativa quanto ao funcionamento das estruturas de indexação (ou métodos de acesso) gira basicamente em torno de dois aspectos: eficiência no tempo e eficiência no espaço. Quanto ao primeiro, o que se espera de um método de acesso é um desempenho característico de uma estrutura convencional, como a "B-Tree"(Seção 4.1.1.2). Por sua vez, para alcançar o segundo, um método de acesso precisa ser pequeno, em termos de espaço ocupado em comparação com os dados que endereça e, uma vez definido o espaço, deve garantir uma certa utilização do mesmo. Entretanto, a "eficiência em tempo e espaço de um método de acesso depende fortemente dos dados a serem processados e das consultas a serem feitas" [Gaede & Gunter, 1998].

Diversas pesquisas realizadas na área de banco de dados para imagens resultaram em uma variedade de métodos de acesso multidimensionais para permitir a execução das várias consultas inerentes a cada tipo de aplicação e para tornar os SRICs realmente escaláveis para grandes coleções de imagens.

Podemos dividir os métodos de acesso em unidimensionais e multidimensionais. As estruturas unidimensionais utilizam um ou mais atributos unidimensionais para a construção do método de acesso, enquanto que as estruturas multidimensionais utilizam atributos multidimensionais.

4.1.1 – Métodos de Acesso Unidimensionais

Os métodos de acesso unidimensionais clássicos são fundamentos básicos dos métodos de acesso multidimensionais discutidos a seguir. As estruturas unidimensionais mais comuns incluem *linear hashing* e Árvores B *(B-Tree)*. Métodos hierárquicos como as árvores B são escaláveis e se comportam bem nos casos de dados de entrada dispersos; eles são quase independentes dos dados de entrada. Isto não é

necessariamente verdade para as técnicas *hashing*, cujo desempenho pode degenerar dependendo dos dados de entrada e da função de *hashing*.

4.1.1.1 – *Hashing* Linear

No método *hash*, uma estrutura de armazenamento é dividida em *k* intervalos chamados *buckets* (uma coleção de registros). Cada *bucket* é identificado com um valor numérico de 0 até *k*. Ao inserir uma informação no sistema, uma função, chamada de função *hash*, é aplicada sobre os valores de alguns atributos (a chave de busca) o que resulta em um valor também entre 0 e *k*, indicando assim, em qual *bucket* a informação será armazenada. De forma similar, um índice *hash* organiza as chaves de busca em uma estrutura *hash*. Cada chave de busca possui um ponteiro associado à posição física das informações na estrutura de armazenamento (memória primária, secundária ou terciária). Na entrada de uma nova informação a mesma função *hash* é aplicada sobre os valores da chave de busca, sendo possível endereçar diretamente o *bucket* onde a nova informação será inserida. No momento de uma consulta, a função endereça o *bucket* que contém a informação desejada, [Silberschatz et al., 1997].

Cada *bucket* tem capacidade de armazenar uma quantidade limitada de registros. Caso esta capacidade seja excedida (situação de *overflow*), o intervalo é novamente dividido em mais *buckets* e os dados reorganizados.

Uma das vantagens do *Hashing* é que ele permite ao sistema de banco de dados determinar diretamente aonde se encontra determinado dado, enquanto que um índice força o sistema a pesquisar uma estrutura de dados, significando uma demora na localização da informação.

4.1.1.2 – Árvores B

Para reduzir o tempo de busca de um índice, é preciso reduzir o número de acessos a dados que o índice usa durante a pesquisa. Isto pode então ser realizado

usando um índice baseado em árvores que armazena os elementos de dados em seus nós-folhas [Silberschatz et al., 1997] A estrutura de índices *B-Tree* organiza os dados de maneira hierárquica. Esta habilidade para ordenar os dados é uma das vantagens da indexação em relação a outras técnicas de acesso a dados, como o *Hashing*.

As árvores B indexam relações através de suas chaves primárias. Sua estrutura permite inclusões e exclusões enquanto permanece balanceada. A Figura 8 apresenta um exemplo de uma estrutura *B-Tree*.

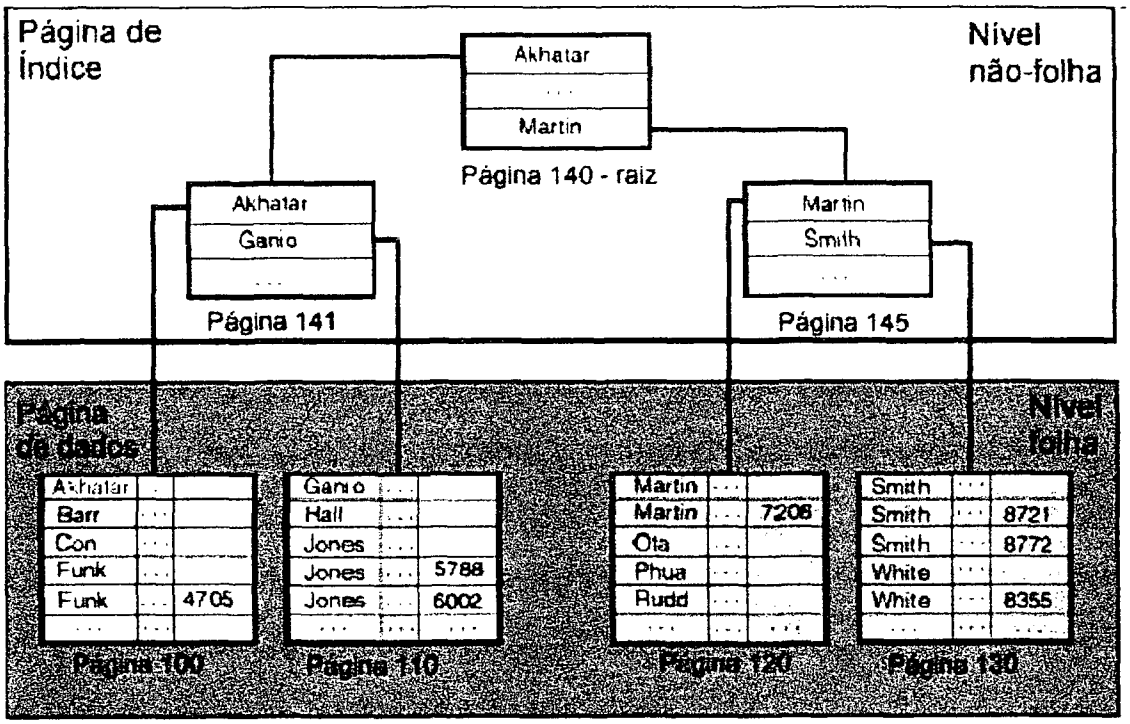


Figura 8: Exemplo de uma árvore B ("B-Tree").

Porém, para a indexação de dados multidimensionais ela não é indicada porque sua estrutura utiliza uma chave simples para indexar os dados. Para usar uma estrutura *B-Tree* com dados multidimensionais, os dados teriam que ser convertidos a uma única dimensão. Um dos métodos que utiliza esta idéia consiste em concatenar todos os atributos em único conjunto de caracteres (*string*) de uma dimensão para cada objeto. Um dos problemas desta técnica é que a dimensão usada para ordenar este grande *string* de características terá prioridade sobre as demais, para se determinar a

distância entre dois objetos dentro do espaço multidimensional, [Brown & Gruenwald, 1998].

Assim, o que os pesquisadores tentam desenvolver é uma estrutura baseada em árvore que seja tão efetiva com dados de imagens como a *B-Tree* é para a indexação de dados tradicionais. Com isso, os métodos multidimensionais tentam modificar as técnicas utilizadas em uma *B-Tree* para torná-las eficientes para sua utilização em bancos de dados de imagens.

4.1.2 – Métodos Multidimensionais

Como uma árvore de índice multidimensional também se baseia na mesma estrutura de dados que uma B-Tree, elas compartilham características similares. Cada nó da árvore de índice corresponde a uma seção específica do espaço multidimensional e cada um dos nós-folhas corresponde a uma específica subseção. Além disso, cada nó armazena um conjunto de valores e ponteiros, assim como ocorre em uma B-Tree.

O conjunto de valores que cada nó da árvore armazena, identifica a região do espaço multidimensional que corresponde ao nó. Por exemplo, se o nó representa uma região retangular do espaço de dados, os valores representariam os limites, inferior e superior, ao longo de cada dimensão.

O conjunto de ponteiros refere-se aos filhos do nó. Se o nó é um *nó-folha*, este ponteiro se refere ao elemento de dados atual, [Brown & Gruenwald, 1998].

A inclusão ou exclusão de um nó requer que a árvore seja reorganizada. Para esta função são utilizadas duas operações: Dividir (*split*) e Mesclar (*merge*),[Date, 1984]. Estas operações podem se propagar por toda a árvore, dependendo do caso a propagação da operação pode ser para baixo(em direção Às folhas) ou para cima (em direção à raiz).

Segundo [Gaede & Gunter, 1998], existe uma pequena divisão entre os métodos de acesso multidimensionais. São os métodos de acesso a pontos (PAMs – *point access methods*), que consideram cada imagem como sendo apenas um ponto no

espaço multidimensional, e os métodos de acesso espaciais (SAMs – *spatial access methods*) que consideram cada imagem como sendo um conjunto de objetos (linhas, círculos, retângulos, polígonos, etc.) no espaço multidimensional.

4.1.2.1 – k-D-B Tree

Esta estrutura de indexação utiliza as características de uma *k-d tree* [4], que permite expandir o número de *nós-folhas* para cada nó interno, unidas às características de uma B-Tree, para reduzir a altura da árvore de índice, [Silberschatz et al., 1997].

Cada um dos nós internos armazena valores para identificar uma seção do espaço multidimensional e um conjunto de ponteiros referencia a seus nós filhos. As seções identificadas por um nó não se sobrepõem com nenhuma outra região identificadas por seus nós irmãos, e as seções são criadas pela divisão das seções maiores correspondentes a seus pais. Cada divisão ocorre em uma única dimensão.

As divisões em uma K-D-B-Tree são realizadas de modo a dividir os pontos de dados em regiões o mais uniformemente possível e também minimizando o número de divisões. Assim, cada região é dividida independentemente de outras partições no espaço de dados. Isto significa que outras partições não devem ser consideradas quando uma região é dividida.

A Figura 9 ilustra a divisão de um espaço de dados de uma árvore K-D-B usando alternativamente as dimensões. Neste exemplo, as letras representam pontos no espaço multidimensional (imagens) e os números as seções formadas pela divisão do espaço.

---

[4] A k-d tree (*k-dimensional tree*) foi uma das primeiras estruturas usadas para indexar múltiplas dimensões, [Silberschatz et al., 1997].

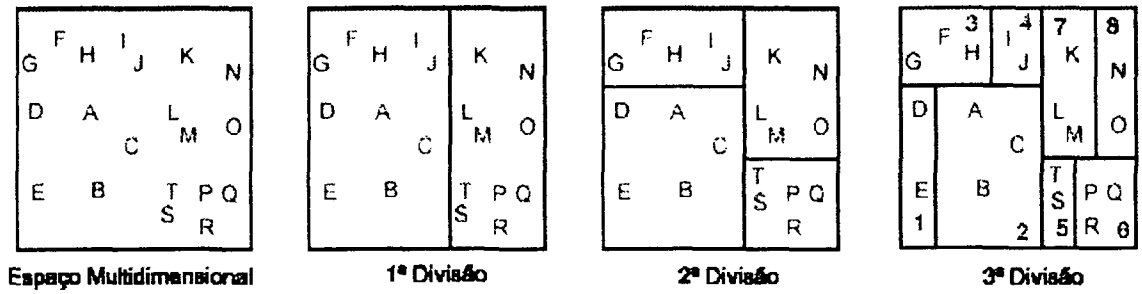


Figura 9: Divisão do espaço de dados por uma árvore K-D-B.

Podemos notar que a primeira divisão separa o espaço de dados verticalmente em duas seções ("1234" e "5678") usando uma dimensão. Na segunda divisão usa-se a outra dimensão, distribuindo uniformemente os dados: na região esquerda "12" e "34" e na direita "56" e "78". A terceira divisão define a separação final das as seções. Estas são as regiões que serão armazenadas em uma árvore B, conforme mostra a Figura 10.

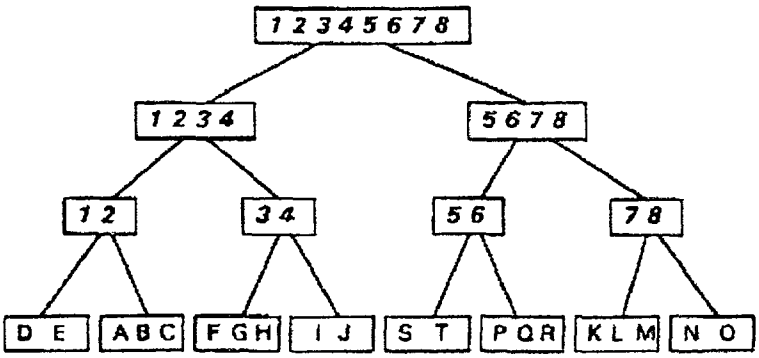


Figura 10: Estrutura de uma Árvore K-D-B

4.1.2.2 – G Tree

Na árvore G-Tree as propriedades de uma estrutura *B-Tree* são combinadas com as propriedades de um *grid file*,[Brown & Gruenwald, 1998]. O *grid file* é um método de acesso a dados que divide o espaço de endereços ao longo de cada dimensão, [Samet, 1994]. O arquivo é assim chamado porque as divisões ocorrem como uma estrutura de grade. Cada vetor de característica nesta estrutura de dados é

então mapeado para o ponto da grade multidimensional mais próxima no espaço de endereços.

A árvore G é uma estrutura de índice balanceada que, como a K-D-B Tree, divide o espaço de dados em um conjunto de regiões retangulares e não sobrepostas. Quando a divisão ocorre em um nó que está completo, a região correspondente é dividida ao meio. Porém numa árvore K-D-B, isto não ocorre, pois a árvore K-D-B divide uma região com o objetivo de obter um número uniforme de pontos de dados em cada região.

A divisão na G-Tree ocorre ao longo de uma simples dimensão e as dimensões usadas se alternam uma após a outra, evitando que uma tenha maior destaqe do que a outra. Uma vez que as dimensões se alternam de forma ordenada, uma única *string* pode ser usada para identificar cada partição. Pelo fato de cada divisão criar duas novas partições, o identificador pode ser uma *string* binária.

Além disso, como uma região é sempre formada pela divisão de uma outra ao meio, o valor usado para efetuar a divisão não precisa ser armazenado. Isto significa que a posição de cada nó em uma árvore G diretamente identifica sua região correspondente. Embora seu procedimento de divisão seja mais restritivo que uma árvore K-D-B, as árvores G requerem menos espaço de armazenamento. A Figura 11 ilustra o espaço de dados dividido por uma árvore G.

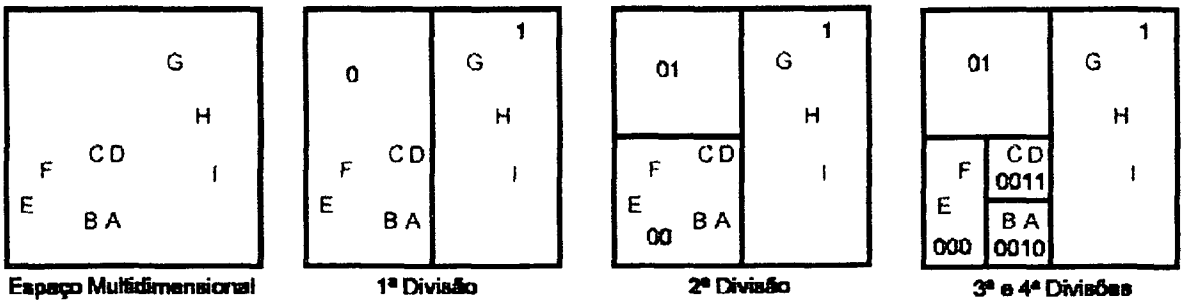


Figura 11: Divisão do espaço de dados por uma árvore G.

Os pontos de dados são identificados por letras e as regiões não vazias são armazenadas em uma árvore B. Estas regiões são, entretanto, identificadas com *strings* binárias correspondendo a sua localização. Por exemplo, a primeira divisão gera a

região de dados a esquerda que é identificada começando com 0 (zero), enquanto que a da direita é identificada começando com 1 (um). Similarmente, na segunda divisão, a parte de cima é identificada começado com 01 e a metade de baixo com 00. As divisões seguem conforme ilustra as demais divisões.

4.1.2.3 – MB^+ Tree

A árvore MB^+ é assim nomeada porque é uma modificação de outra variação comum da árvore B, chamada B^+ Tree,[Brown & Gruenwald, 1998]. Em uma árvore B^+, os valores de dados são armazenados nas folhas e copiados no nó interno na árvore se necessário. As folhas são agrupadas de forma que elas possam ser pesquisadas rapidamente.

Como as árvores G, as árvores MB^+ dividem o espaço de dados em várias regiões retangulares disjuntas. Também como as árvores G, as regiões são ordenadas e a árvore é balanceada.

Neste método o espaço de dados é dividido em diversas fatias ao longo da primeira dimensão. E estas divisões ocorrem até que as *tiras* resultantes se tornem muito estreitas, ou seja, a largura é menor do que um valor predeterminado. Quando isso ocorre, as divisões do espaço de dados ocorrem independentemente das outras regiões ao longo da segunda dimensão. As regiões continuarão a serem divididas desta maneira até que as tiras se tornem também estreitas com relação a segunda dimensão. Neste ponto, a próxima dimensão é dividida. Este padrão continua quando novos elementos são inseridos à árvore.

Assim uma das vantagens da árvore MB^+ é que, em contraste com as árvores G, as fatias ao longo de uma dimensão podem ocorrer em qualquer lugar. Assim, se existir um número muito grande de pontos de dados agrupados em uma região, a árvore MB^+ pode dividir o número de pontos de dados uniformemente. Isto diminui o número de níveis em uma árvore, e assim reduz o tempo gasto pela busca de elementos

de dados. A desvantagem é que o valor usado para a divisão deve ser agora armazenado a cada nível.

A Figura 12 mostra um espaço de dados bi-dimensional dividido por uma árvore MB^+. Novamente, os pontos de dados são identificados por letras. As regiões de dados serão, entretanto armazenadas em uma árvore B. Estas regiões são identificadas usando a ordem das dimensões. Por exemplo, as regiões de dados na esquerda são identificadas começando com 0 (zero), as identificações do meio começam com 1 e as da direita começam com 2. O segundo número indica a posição da região ao longo da próxima dimensão.

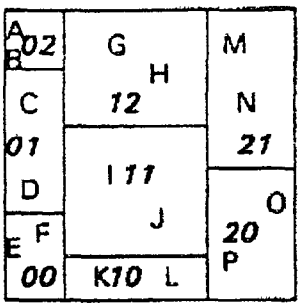


Figura 12: Divisão do espaço de dados por uma árvore MB^+.

Estes identificadores podem ser usados para todas as regiões de dados. Com esta simples ordenação, as regiões de dados são organizadas em uma árvore B. Isto permite ao algoritmo de pesquisa mover diretamente de uma região para outra usando a lista unida de folhas.

4.1.2.4 – BV Tree

A árvore BV foi projetada para superar uma das desvantagens da árvore K-D-B que ocorre quando a inserção de um elemento em uma partição do espaço de dados causa a divisão do nó correspondente,[Brown & Gruenwald, 1998]. Especificamente, a divisão pode se propagar para baixo, significando que todos os seus nós filhos tenham que ser divididos também ao longo da mesma dimensão usada para

dividi-lo. Esta é uma grande desvantagem porque a divisão que separa os nós de elementos de dados uniformemente pode não dividir seus filhos uniformemente.

Para resolver esta situação, a árvore BV usa o conceito de promoção do espaço de dados durante a divisão. Uma partição é promovida pela movimentação de seu nó corrente na árvore BV para seu nó-pai. Isto é feito se uma das partições do espaço de dados desenvolvida de uma divisão completa está completamente contida dentro de outra partição abaixo dela. A menor delas deve ser promovida para o nível da partição mais alta e é referenciada como seu *protetor*. Assim, ao invés dessa operação de divisão se propagar para as partições menores, ela é simplesmente promovida.

Um exemplo de promoção é mostrado na Figura 13. Considerando que a partição 7 é um subconjunto da partição 5, a partição em destaque. A partição 5 então, deve ser promovida com seus filhos tal que ela está no mesmo nível que a partição 7. Depois desta promoção, a partição 5 é o *protetor* da partição 7.

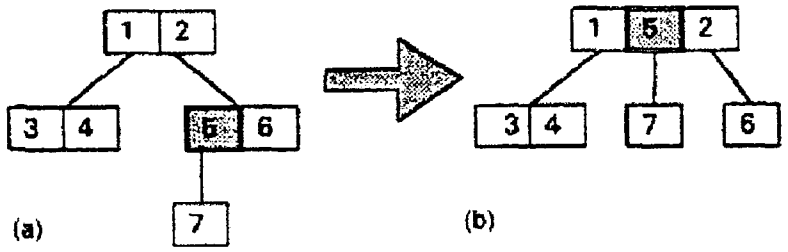


Figura 13: (a) Estrutura de uma árvore BV.(b) Promoção de um nó em uma árvore BV.

Uma das desvantagens que surgem da promoção de protetores é que os algoritmos de busca definidos para as árvores BV são mais complexos. Embora promovidos, os protetores devem ser pesquisados como se eles estivessem em suas posições originais na árvore.

4.1.2.5 – hB Tree

Assim como a árvore BV, a árvore hB ou "*holey-brick B Tree*" foi criada para superar uma desvantagem surgida da divisão de um nó completo de uma árvore

K-D-B,[Brown & Gruenwald, 1998]. Para prevenir que a divisão se propague para baixo em um nó filho, as árvores hB usam o conceito de "*holey-bricks*". Esta estrutura recebe este nome porque se refere às regiões que são formadas quando o espaço de dados é dividido. Para dividir um espaço de dados, as partições das regiões podem ocorrer ao longo de mais de uma dimensão. Sendo possível criar áreas retangulares na região pela partição ao longo de pedaços de vários tamanhos. Isto permite mais flexibilidade em como os dados serão divididos. Esta flexibilidade provê uma solução para as situações onde cortando através da região de um nó pela dimensão inteira significará dividir as sub-regiões filhas.

Isto não apenas previne a propagação da divisão para baixo, mas também distribui os elementos de dados mais uniformemente em situações onde um corte em uma simples dimensão não poderia faze-lo. Para ilustrar uma situação dessas, a Figura 14a possui um espaço bi-dimensional com os pontos de dados dispostos como um sinal de adição (+). Para dividir os dados mais uniformemente, uma divisão bi-dimensional de uma região retangular deve ser usada, conforme ilustra a Figura 14b.

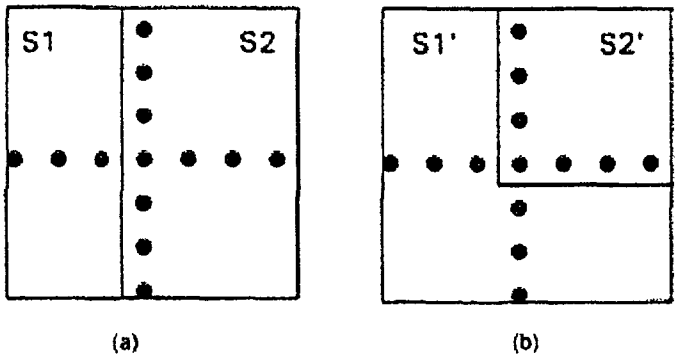


Figura 14: Espaço de dados dividido por uma árvore K-D-B (a) e por uma hB (b).

A desvantagem deste esquema de indexação é que requer um maior espaço para armazenar os nós. Isto porque um nó pode agora corresponder a uma região retangular do espaço de dados.

4.1.2.6 – VP e MVP Trees

Embora a divisão em muitas árvores seja baseada na posição dos dados ao longo de uma dimensão, a árvore "*vantage point*" (VP) usa uma técnica diferente,[Brown & Gruenwald, 1998]. A cada nó da árvore ela seleciona um dos pontos de dados para a função de ponto favorável ("*vantage point*") e a divisão dos pontos de dados restantes é baseada na distância entre o ponto favorável e os demais pontos do espaço de dados. A divisão é feita para que o número de pontos de dados em cada região seja o mais uniforme possível. Pelo fato dela ser baseada na distância em relação a um ponto, as regiões de uma árvore VP são esféricas ao invés de retangulares. Conforme mostra a Figura 15a.

Uma das vantagens da árvore VP é que ela é naturalmente adaptada para consulta do tipo "vizinho mais próximo" (*nearest neighbor*). Uma desvantagem, entretanto, é que quando o espaço de dados tem um número muito grande de dimensões, as partições começam a ficar muito estreitas. Isto aumenta o número de ramos da árvore que devem ser pesquisados.

Por causa desta desvantagem, uma modificação da árvore VP foi proposta e chamada de MVP Tree ("*multi-vantage-point*") Figura 15b. Ao invés de um ponto favorável, a árvore MVP usa dois. O primeiro ponto é usado para criar *m* partições, onde *m* é a ordem da árvore. Em cada uma dessas partições, o segundo ponto é usado para criar mais *m* partições para um total de m^2 partições. A vantagem é que se pode criar mais divisões por nós internos da árvore, o que diminui o tempo de busca.

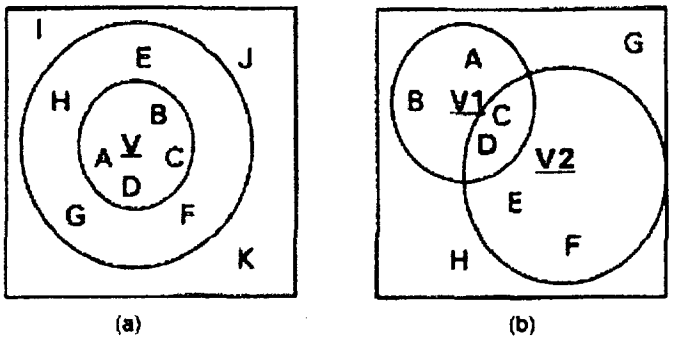


Figura 15: VP Tree (a) e MVP Tree (b).

4.1.2.7 – LSD Tree

A árvore LSD, ou "*local split decision*", é assim nomeada porque o critério usado para dividir o espaço de dados é executado independentemente para cada partição retangular,[Brown & Gruenwald, 1998]. A divisão não é restrita a uma dimensão específica ou se ela divide ou não o espaço de dados pela metade. Isto significa que podemos dividir o espaço de dados em qualquer dimensão e em qualquer valor que escolhermos. A Figura 16a ilustra uma divisão possível.

Esta flexibilidade tem uma desvantagem, entretanto. Desde que qualquer dimensão possa ser usada para o particionamento a qualquer instante, a informação da divisão deve ser armazenada para cada nó. Assim, um diretório é necessário para armazenar as partições. Especificamente, o diretório deve armazenar as dimensões e os valores de dados usados para a divisão.

A Figura 16b ilustra o diretório criado para armazenar as informações da divisão do espaço de dados ilustrado pela Figura 16a.

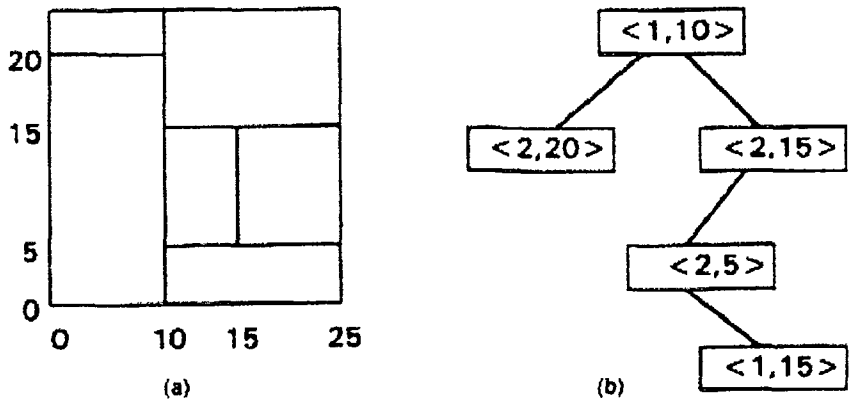


Figura 16: Divisão do espaço (a) por uma árvore LSD e seu respectivo diretório (b).

As estruturas de árvores vistas anteriormente armazenam os elementos de dados como sendo pontos no espaço multidimensional. No caso, métodos de acesso a pontos (PAMs). As árvores que serão descritas a seguir armazenam os elementos de dados como formas geométricas dentro do espaço multidimensional, métodos de acesso espaciais (SAMs) ,[Brown & Gruenwald, 1998].

Para isso elas utilizam o conceito de MBR (*Minimum Bounding Region*), que é a menor região, geralmente retangular, que cobre toda a figura geométrica. A Figura 17 ilustra exemplos de MBR retangular.

Cada nó da árvore descrita na Figura 17 corresponde a MBR de seu grupo de formas. Da mesma maneira que nas árvores vistas anteriormente, quando uma inserção é aplicada a um nó completo, os elementos são divididos em dois grupos. Cada um desses grupos é armazenado em um novo nó contendo sua respectiva MBR, [Theodoridis & Sellis, 1994].

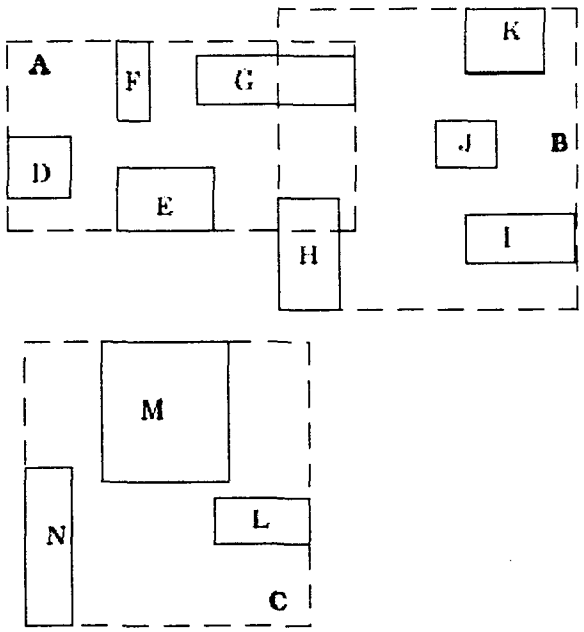


Figura 17: Alguns retângulos organizados e suas MBRs.

4.1.2.8 – R Tree, R+ Tree, R* Tree

A R-Tree proposta por [Guttman, 1984] é uma das mais populares estruturas de dados para indexação de dados espaciais. Nesta árvore, cada nó contém tuplas no seguinte formato <*I* , *ptr*> onde *ptr* é o endereço de um filho do nó, e *I* indica sua MBR retangular, pela identificação dos limites inferior e superior de cada dimensão. Similarmente aos valores de uma árvore B, o número de tuplas em cada nó de uma R-Tree não pode ser maior do que *M* ou menor do que *m*<= M/2.

Da mesma forma que uma árvore B, a árvore R também é balanceada. A Figura 18 ilustra a divisão de um espaço multidimensional ilustrado na Figura 17 por uma árvore R.

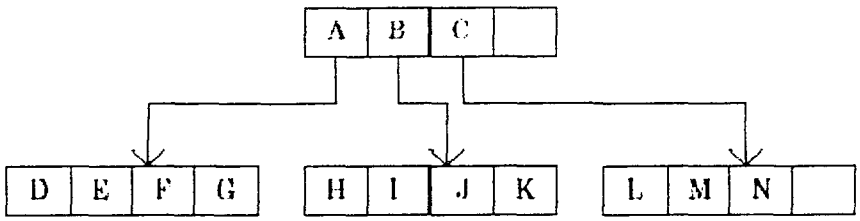


Figura 18: Árvore R para as MBRs da Figura 17.

Dois conceitos importantes que são avaliados nas árvores R-Tree são: o conceito de cobertura, que é definido como a área total de todos os retângulos associados com os nós daquele nível. E, sobreposição (*overlap*), que é definido como a área total contida em dois ou mais retângulos sobrepostos.

Com base nestes dois conceitos, uma R-Tree eficiente deve ter ambos, cobertura e sobreposição minimizados. A diminuição da cobertura implica em menos espaço "*morto*" (espaço vazio) coberto pelos nós. A redução da sobreposição implica em um número menor de ramos da árvore a serem pesquisados durante uma busca.

Na Figura 19, observamos uma janela de busca definida pelo retângulo W. Ambas sub-árvores roteadas pelos nós A e B devem ser pesquisadas, embora apenas a segunda retornará uma resposta satisfatória. O custo desta operação é pelo menos de um acesso a página para a raiz (*root*) da árvore e dois acessos adicionais para verificar os retângulos armazenados em A e B.

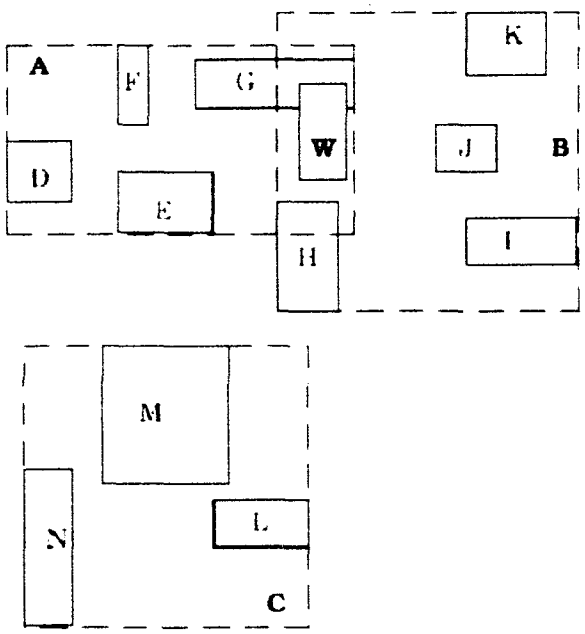


Figura 19: Exemplo de uma janela de busca em uma árvore R.

Com o objetivo de resolver estes problemas nas R-Trees, surgiram então algumas variantes: R^+-Tree [Sellis et al.,1987] e R^*-Tree [Beckmann et al., 1990]. A R^+-Tree difere da árvore R-Tree no fato de que não é permitido que uma MBR se sobreponha a outra. Isto é realizado permitindo aos elementos de dados serem divididos entre diferentes nós na árvore. Quando se permite a sobreposição de MBR, o algoritmo de pesquisa deve "percorrer" múltiplos caminhos dentro da árvore. Se eles não podem se sobrepor, o objeto é localizado usando apenas um ramo da árvore. A desvantagem deste método em proibir a sobreposição é que o algoritmo de exclusão é mais complexo em uma árvore R^+-Tree do que em uma árvore R.

Já árvore R^*-Tree é mais semelhante a R-Tree no fato que permite a sobreposição das MBRs. A diferença entre elas é que a R^*-Tree baseia suas MBRs em suas áreas.

4.1.2.9 – *Buddy Tree*

Como as árvores R^+, a árvore *buddy* previne suas MBRs de sobreposição,[Brown & Gruenwald, 1998]. Ao contrário das árvores R^+, entretanto, seus dados espaciais não precisam ser divididos entre múltiplas rotas. Assim,

pesquisas podem ser executadas usando apenas uma rota na árvore. Esta árvore está apta a realizar isto restringindo a MBRs a regiões chamadas "*regiões companheiras*" ("*buddy regions*"). Retângulos companheiros ("*buddy rectangles*") são formados pela divisão repetida do espaço de dados em duas partes de mesmo tamanho por hiperplanos orientados pelos eixos do espaço de dados. Cada nó interior corresponde a uma partição d-dimensional. O uso deste particionamento previne que a divisão se propague para baixo como ocorre com as árvores R^+.

4.1.2.10 – P Tree

As árvores poliédricas (*polyhedral Trees*) ou *P-Tree*, usam uma técnica diferente para suas MBRs,[Brown & Gruenwald, 1998]. Ao invés de usar uma região limite retangular, a árvore P usa uma forma mais geral. Especificamente, os lados da região limite não têm de ser perpendicular à direção de qualquer dimensão do espaço de dados. Em muitos casos, isto reduz o volume das MBRs. Uma vez que o volume é reduzido, a sobreposição é reduzida e a probabilidade de pesquisar muitos ramos da árvore diminui.

As entradas em um nó de uma *P-Tree* diferem daquelas usadas em uma árvore R, pois suas MBRs devem ser especificadas diferentemente. A MBR de uma árvore R pode ser identificada pelo limite superior e inferior de cada dimensão do espaço de dados. Em contraste a isso, as regiões poliédricas podem ser compostas de hiperplanos dispostos em várias orientações diferentes. Estas orientações definem a orientação do espaço que pode ter mais dimensões que o espaço de dados. A Figura 20 ilustra um exemplo de um espaço de dados dividido em MBRs de uma P-Tree.

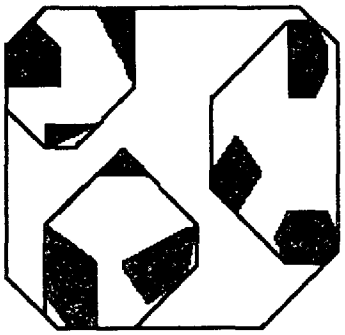


Figura 20: Espaço Dimensional organizado por uma P-Tree.

4.1.2.11 – X Tree

O termo Xtree se refere a "*eXtended node tree* ", [Brown & Gruenwald, 1998]. Esta árvore X também é focada na minimização da sobreposição que ocorre entre as MBRs, para reduzir a quantidade de rotas da árvore que devem ser pesquisadas. A árvore não faz, entretanto, a divisão do espaço de dados em múltiplas MBRs como uma *R-Tree*. Ao invés disso, esta árvore analisa a quantidade de sobreposição que irá ocorrer para cada divisão, então escolhendo a menor sobreposição ou criando um *super-nó* que a evite totalmente, conforme mostra a Figura 21. Isto envolve um algoritmo mais complexo usado na divisão do que uma *R-Tree* original.

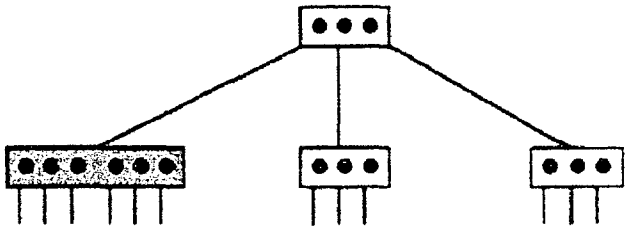


Figura 21: Exemplo de um super-nó da X-Tree

O termo *super-nó*, refere-se a um nó interno da árvore X, no qual a capacidade é maior do que nos nós normais. Isto significa que ele pode conter mais valores de dados e ponteiros do que o limite máximo permitido para os outros nós. Isto

também auxilia na redução do número de níveis de uma árvore, o que melhora o tempo de pesquisa para localizar todos os nós.

4.1.2.12 – SR Tree e SS Tree

A árvore SR ("*sphere-rectangle*") é uma modificação da árvore SS ("*sphere-sphere*"), [Brown & Gruenwald, 1998]. Em uma árvore SS os nós correspondem a MBR esférica dos objetos de dados. A principal vantagem do uso de esferas ao invés de retângulos é que a anterior pode ser representada por apenas seu centro e seu raio. Isto requer menos espaço do que retângulos que requerem o armazenamento dos valores de limite superior e inferior da estrutura de cada dimensão. A Figura 22 apresenta uma representação de uma SS-Tree.

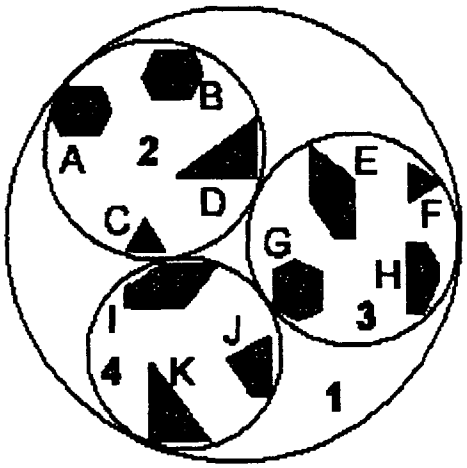


Figura 22: Exemplo de espaço organizado por uma SS-Tree

Uma desvantagem das árvores SS é que uma MBR esférica tem mais volume de cobertura do que uma MBR retangular. Isto aumenta a sobreposição. Este problema é diminuído pela árvore SR, onde cada nó corresponde à interseção da MBR esférica com a MBR retangular de seus elementos de dados. Enquanto isto reduz a quantidade de sobreposição, o uso da interseção de duas formas geométricas significa que uma representação mais complexa é necessária para cada nó. Especificamente, esta região é identificada pelo centro de sua MBR esférica e a menor das maiores distâncias às MBRs esféricas e retangulares de seus filhos.

Esta complexa representação significa que mais recursos computacionais são necessários para computar estas regiões, os quais aumentam o tempo gasto com inserções e exclusões. A Figura 23 ilustra uma árvore SR. Pode-se observar que em uma SR Tree a área sombreada é a área coberta pelo nó, de forma que o espaço "*morto*" é menor do que em uma SS-Tree.

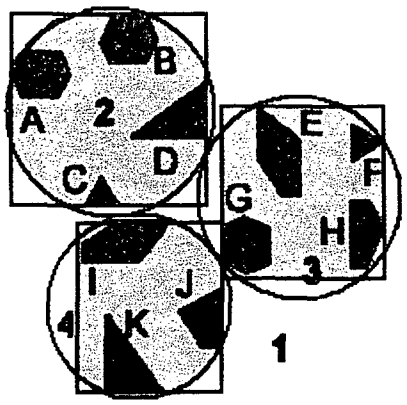


Figura 23: Exemplo de espaço organizado por uma SR-Tree.

4.1.2.13 – TV Tree

O grande problema da maioria das árvores vistas até o momento é que elas são ineficientes quando trabalham com dados que têm um número muito grande de dimensões. Por causa disso, a árvore "*telescopic vector*" foi desenvolvida. Seu objetivo é reduzir o número de dimensões utilizadas por um índice, [Brown & Gruenwald, 1998].

Esta redução é realizada pelo uso de dimensões que são realmente necessárias para distinguir pontos de dados. Tais dimensões são chamadas *ativas*. Isto é oposto às dimensões *inativas* que descrevem as dimensões no começo do vetor de características e que tem os mesmos valores. Determinar quais as dimensões que são necessárias para distinguir os pontos de dados implica que algumas características são mais importantes que outras.

Quando os dados contidos em uma MBRs de uma árvore mudam, o número de dimensões ativas também muda. O número da dimensão ativa, entretanto, permanece constante. Assim uma árvore TV-α é aquela que contém α dimensões ativas. Por exemplo, uma MBR de uma árvore TV-2 contém os pontos: <3, 0.5, 5, 3>, <3, 0.5, 7, 0> e <3, 0.5, 10, 8> . Desde que os valores na primeira dimensão dos pontos de dados são os mesmos, as dimensões 1 e 2 são inativas. As dimensões ativas são as dimensões 3 e 4. Isto significa que na inclusão de pontos de dados específicos de uma MBR, cada nó deve identificar o número da dimensão que ele armazena, com o último α considerado ativo.

4.1.2.14 – VAMSplit R Tree

Esta estrutura é uma variante da estrutura R-Tree. É criada recursivamente escolhendo-se a divisão do conjunto de dados usando a dimensão de máxima variância e escolhendo uma divisão que é aproximadamente a média dos valores de dados [White & Jain, 1996a]. Desde que o critério de divisão é dependente da dispersão dos dados no lugar do número de dados em um *bucket*, ela provê uma maior efetividade, tanto para dados agrupados quanto para dimensões correlatas. Uma árvore VAMSplit R pode ser construída com duas ordens de magnitude mais rapidamente do que uma árvore R. O objetivo básico deste algoritmo é fornecer mais informações sobre os limites inferiores e superiores das MBRs para o algoritmo de divisão do espaço dados de uma R-Tree.

4.1.2.15 – *M-Tree e Slim-Tree*

A árvore Slim [Traina Jr et al., 2002] é uma modificação da árvore M [Ciaccia et al.,1997]. Ambas são chamadas de árvores métricas pois utilizam uma função de similaridade para determinar a distância relativa entre objetos, e usam esta

distância para organizar e dividir o espaço de pesquisa. Uma função de similaridade *d* deve atender a três regras de um espaço métrico:

- Simetria: $d(x,y) = d(y,x)$, onde *d* é a função de similaridade;
- Não negatividade: $0 < d(x,y) < \infty$, $x \neq y$ e $d(x,x) = 0$;
- Desigualdade triangular: $d(x,y) \leq (d(x,z) + d(z,y))$;

A técnica usada pelas árvores métricas divide o espaço métrico em uma série de partições, onde cada partição possui um elemento central ou representante. Cada representante tem um raio de cobertura (similar às estruturas *VP-Tree* e *MVP-Tree*) e cada um dos objetos, que está dentro deste raio, faz parte da partição e é associado a este representante. Conforme pode ser observado na Figura 24a.

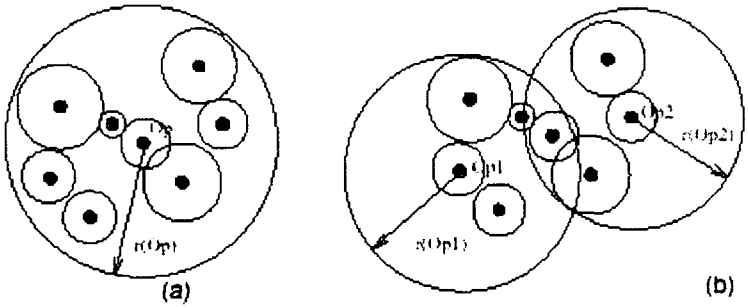


Figura 24: Espaço métrico coberto pelo ponto Op e a divisão do espaço.[Ciaccia,1997]

Em ambas, a árvore é montada de baixo para cima (*bottom-up*), ou seja, quando a inserção de um novo objeto no espaço de dados esgota a capacidade de um nó *N* (*overflow*), um novo nó é criado no mesmo nível do nó *N*, os objetos cobertos por *N* são divididos novamente entre os dois nós (ver Figura 24b) e as informações da criação do novo nó são atualizadas no nível superior de *N (nó-pai)*. Quando isso ocorre na raiz da árvore, uma nova *raiz* é criada e a árvore cresce um nível.

Existem pequenas diferenças entre a *M-Tree* e a *Slim-Tree*. Em [Traina Jr et al., 2002], os autores elaboraram um algoritmo mais eficiente realizar a operação de *split* em uma árvore e um algoritmo com menor custo computacional para orientar a

inserção de um novo objeto na sub-árvore, adequada conforme os dados sejam inseridos no sistema.

Da mesma forma que as *M-Trees*, as *Slim-Trees* não suportam exclusões, em geral, as exclusões são consideradas apenas lógicas, ou seja, a partição é apenas marcada como "excluída", [Traina Jr et al., 2002].

4.1.3 – Conclusões sobre Métodos de Acesso

Em um DBMS convencional o conjunto de tipos de dados embutidos e métodos de acesso é limitado. Além disso, estruturas de indexação *B-Tree* são apropriadas para determinados tipos de dados, conforme discutido anteriormente, e são requeridos freqüentemente métodos de acesso novos para alguns tipos de dados. Por exemplo, árvores de K-D-B e R-Trees [Gutman, 1984] são métodos de acesso apropriados para dados do tipo ponto e polígono respectivamente.

A maioria dos métodos de acesso multidimensionais descritos procura implementar um mecanismo rápido e eficiente em termos de tempo e espaço para realizar a divisão do espaço ou distribuição dos dados e criar uma estrutura de árvore para indexá-lo. O que pode ser verificado é que, entre os métodos de acesso multidimensionais, as árvores R-Tree são as mais utilizadas por causa da facilidade de sua implementação.

Como exemplo, em [Theodoridis & Sellis, 1994] um algoritmo baseado nas árvores R (e suas variantes) é implementado procurando satisfazer aos seguintes critérios:

- minimizar a cobertura de um nó melhorando a precisão na consulta;
- minimizar o *espaço morto* (ou vazio) coberto por um nó, diminuindo o tempo de busca;
- minimizar a sobreposição entre os ramos da árvore, diminuindo assim o número de ramos a serem percorridos durante uma pesquisa;

Este método procura fazer a divisão do espaço dimensional de acordo com os dados existentes. Assim, é um método que depende destes dados (*data-driven*). Para uma outra situação o método pode ter um desempenho diferente do encontrado. A diferença, em termos de desempenho, não é muito grande com relação aos outros métodos (R, R^+ e R^*).

Em [Alexandrov et al., 1995] uma implementação de R Tree é usada combinada com técnicas de avaliação da importância de determinadas características na busca por um determinado padrão de textura. São propostos três conjuntos diferentes de características para a indexação baseada na "*melhor característica*", na "*característica média*" e uma combinação de ambas. E isso facilita o desenvolvimento eficiente de um mecanismo de indexação.

Em [Gaede & Gunter, 1998] são apresentados alguns estudos comparativos realizados com alguns dos métodos vistos, além de outros não citados neste trabalho. Este estudo reafirma que não existe ainda uma ferramenta capaz de identificar se existe um método que seja o mais eficiente de todos. Porém, apresenta uma lista descrevendo os métodos mais eficientes segundo alguns critérios.

Os critérios usados para avaliar o desempenho destas estruturas dizem respeito à quantidade de espaço utilizado para armazenar as informações da árvore, o número de acessos aos nós na busca por determinado objeto e a quantidade de recursos computacionais utilizados na manutenção da árvore (inclusões e exclusões).

Existem, porém, uma série de algoritmos de otimização para estas estruturas, indicando que outros fatores podem interferir no desempenho do sistema, tais como:

- a quantidade de imagens armazenadas no banco de dados;
- a ordenação das características dentro do vetor;
- a dispersão ou agrupamento dos dados dentro do espaço multidimensional;

Em [Kornacker et al., 1999], os autores propõem um conjunto de itens a serem verificados para avaliar a eficiência de uma estrutura de árvore. É apresentada,

também, uma ferramenta gráfica para a visualização da estrutura criada e da distribuição dos dados pela estrutura.

Depois das análises realizadas, convém enumerar as principais características que um método de acesso multidimensional deve ter, baseado nas propriedades dos dados espaciais e sua aplicação conforme sugere [Gaede & Gunter, 1998]:

Dinâmica: Como os objetos de dados são inseridos e excluídos do banco de dados em qualquer ordem, o método de acesso deve continuamente manter registradas as alterações;

Gerenciamento do Armazenamento Secundário e Terciário: Apesar do aumento da capacidade da memória principal, não é possível armazenar todo o banco de dados na memória principal. O método de acesso precisa prover a integração dos dispositivos de armazenamento secundário e terciário da mesma maneira;

Grande variedade de operadores suportados: Métodos de acesso não devem suportar apenas um tipo particular de operação;

Independência dos dados de entrada: Métodos de acesso devem manter sua eficiência mesmo quando os dados de entrada são distribuídos diferentemente ao longo de várias dimensões;

Simplicidade: Métodos de acesso complicados com casos especiais são geralmente propensos a erros ao serem implementados e não possuem robustez suficiente para ser usados em aplicações de larga escala;

Escalabilidade: Métodos de acesso devem adaptar-se ao crescimento inerente do banco de dados utilizados;

Eficiência no tempo: Pesquisas espaciais devem ser rápidas. O principal objetivo do projeto é atingir um desempenho característico de uma B-Tree unidimensional: Primeiro, o método de acesso deve garantir, para o pior caso (*worst case*), uma performance O(log N), onde N é o número de imagens no banco de dados, para todas as possíveis entradas independente da ordem em que forem inseridas.

Segundo, este mesmo pior caso deve ser idêntico para qualquer combinação de atributos;

Eficiência de espaço: Um índice deve ser pequeno se comparado com o dado a ser endereçado e uma vez definido o espaço necessário para armazená-lo, deve garantir uma certa utilização do mesmo;

Concorrência e Recuperação: Nos bancos de dados modernos onde múltiplos usuários concorrentemente atualizam, recuperam e inserem dados o método de acesso deve ser capaz de prover técnicas robustas de controle transacional sem penalizar significativamente o desempenho;

Impacto mínimo: A integração de um método de acesso a um banco de dados deve ter um mínimo impacto nas partes existentes do sistema;

Neste trabalho, inicialmente pretende-se que a estrutura de indexação seja mais eficiente que uma leitura completa das imagens no sistema (*full table scan*). A taxa de utilização ou espaço utilizado pelo índice não será uma preocupação na fase inicial uma vez que o objetivo é reduzir o número de leituras físicas. Neste sentido, [White & Jain, 1996a] afirmam que a idéia comum das estruturas de indexação é utilizar uma hierarquia que seja capaz de eliminar rapidamente objetos que não fazem parte do espaço de busca e aplicar um critério mais refinado, conseqüentemente de maior custo computacional, quando necessário. Assim sendo, as estimativas e medições de custo de processamento serão utilizadas apenas para questões informativas. E os critérios no projeto da estrutura de indexação serão: eliminar rapidamente falsas candidatas e refinar a busca com as *possíveis candidatas*.

5. MECANISMOS DE CONSULTAS

O mecanismo de consulta de um banco de dados é o responsável por identificar as tuplas que respondem ao critério de pesquisa. Para acessar um dado (imagem), representado por um vetor de características, qualquer uma das dimensões pode ser usada. "Porém, para que um índice seja eficiente na recuperação de uma imagem, é necessário que ele seja capaz de pesquisar em todas as dimensões dos dados", [Brown & Gruenwald, 1998].

5 .1 – TIPOS DE CONSULTAS

Segundo [Brown & Gruenwald, 1998], a estrutura de indexação de um banco de dados usado em um SRIC deve estar apta a satisfazer eficientemente a três tipos de consultas:

5.1.1 – *Range-query*

Este tipo de consulta solicita que o sistema retorne todos os elementos que estão contidos ou cruzam com uma região específica dentro do espaço multi-dimensional. Caso a consulta busque um conjunto particular de características ela é chamada de *Subpattern matching query*. Pois apenas alguns atributos são usados para realizar a pesquisa. Caso a consulta seja mais restritiva e se aplique a uma região pequena que possa abranger apenas um ponto do espaço multidimensional então ela é chamada de *Exact point* ou *point query* , pois ela busca por um elemento específico.

Exemplo: Aplicando uma consulta pelos elementos que possuem valor para a dimensão X entre 0 e 2, para a dimensão Y entre 1 e 2 e para a dimensão Z entre 0 e 5. A região no espaço multidimensional definido por esta consulta pode ser visualizada na Figura 25.

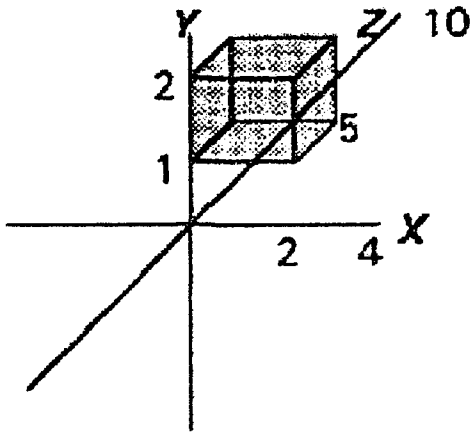


Figura 25 Exemplo de uma consulta por Faixa de Valores (range-query).

5.1.2 – *Nearest neighbor query*

Descrevem consultas nas quais um usuário requisita elementos que são similares a uma região específica do espaço multidimensional. Isto implica que o sistema de Recuperação de Imagens tenha o conceito de similaridade implementado. Na Figura 26, abaixo, temos um exemplo de um sistema que usa a distância Euclideana para determinar a similaridade entre elementos. Um usuário que procure pelos elementos similares ao elemento do ponto <3, 0.5, 4>, receberia como resultado os pontos <2, 1,7> e <4, 1.5,8>.

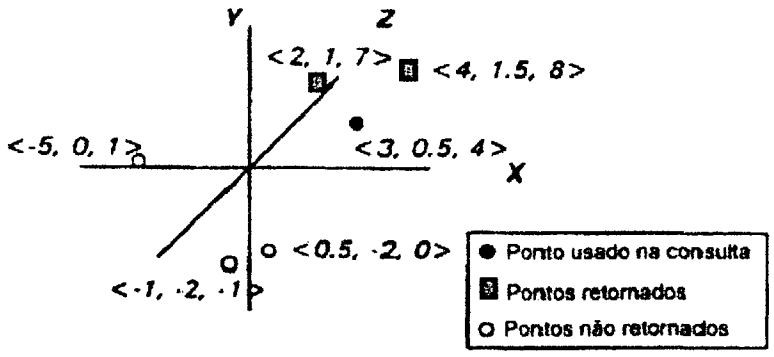


Figura 26: Consulta "nearest neighbor".

5.1.3 – *Spatial Joint Query*

Consultas que retornam pares de elementos similares. Um exemplo deste tipo de busca seria para encontrar elementos redundantes em um banco de dados. Este tipo de consulta também exige que o sistema possua o conceito de similaridade implementado. Para atender este tipo de consulta o sistema necessita também de uma função que receba as imagens como valores de entrada e retorne um valor escalar como saída. Este valor, chamado *distância*, é então usado para determinar a similaridade entre os pontos. O objetivo deste método é encontrar pares de imagens cuja distância seja menor que um determinado valor. A Figura 27, abaixo exemplifica uma consulta deste tipo onde a distância Euclideana é menor do que 2.

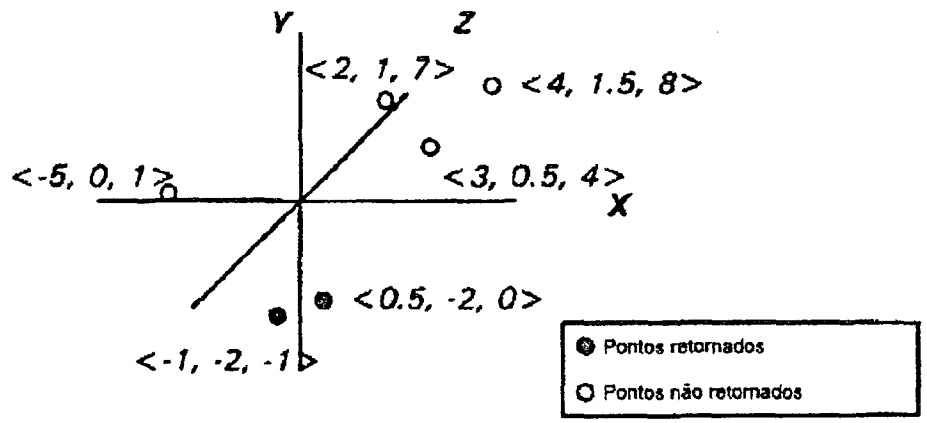


Figura 27: Consulta do tipo "spacial join".

Índices apropriados são necessários para eficientemente resolver estas consultas. Então vejamos.

- *Range query*: o índice deve ordenar os dados;
- *Nearest neighbor* : o índice deve manter informações sobre a distância entre os elementos, e deve mover para o vizinho mais próximo de um determinado ponto usando o índice;
- *Spatial join*: o índice deve auxiliar na união espacial agrupando pontos que estão próximos um do outro;

6. SRIC 3D

O SRIC3D é um Sistema de Recuperação de Imagens por Conteúdo 3D, atualmente em desenvolvimento pelo Grupo IMAGO da UFPR[5]. Este projeto usa imagens de profundidade (*range images*) como base de dados para a reconstrução de modelos 3D a partir das vistas de objetos. Na fase atual do projeto, estão sendo realizados estudos de algoritmos para a extração de características e mineração de imagens. O esquema geral do foi apresentado na Figura 1 na Introdução deste trabalho.

A fase *Aquisição* corresponde ao cadastramento das imagens e de suas informações no banco de dados. Para esta etapa do projeto, foram utilizadas algumas imagens da base de dados OSU (MSU/WSU) Range Image Database[6], que possui cerca de 700 objetos, onde para cada objeto existem várias vistas. Outras imagens estão sendo adquiridas com um *scanner* 3D (um *MODELA MDX-15 touch-probe*)[7] mas o projeto prevê o recebimento de imagens de outras bases disponíveis na Internet.

Para transformar cada uma das características a serem manipuladas em dados formais, é preciso aplicar sobre a imagem um ou mais algoritmos de extração de características, conforme visto no Capítulo 3.

Junto com cada imagem existe um arquivo que contém os dados das bandas *X*, *Y* e *Z*, ou seja, as informações espaciais da imagem e, a partir dele, são extraídas as informações sobre a localização de detalhes e regiões da imagem, bem como seus inter-relacionamentos espaciais. Este arquivo também é cadastrado na base de dados para que, futuramente, outras características também possam ser extraídas a partir deles.

---

[5] www.inf.ufpr.br/imago.

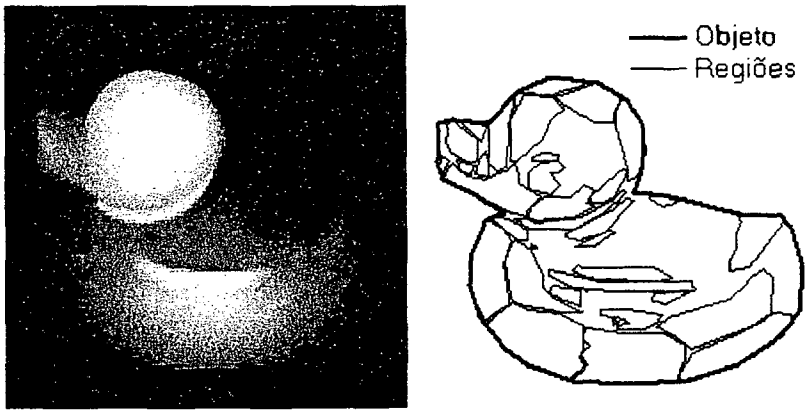
[6] http://sampl.eng.ohio-state.edu/~sampl/data/3DDB/RID/index.htm.

[7] Detalhes em: http://www.roland.netsys.it/.

Para obter as características espaciais é necessário um processo de segmentação e identificação de regiões dos objetos da imagem, processo este, que utiliza um conjunto de algoritmos envolvendo as seguintes atividades:

- poligonização das bordas, gerando as arestas do polígono que formam a borda do objeto;
- segmentação da imagem em regiões, baseada em mapas de curvaturas das superfícies, identificando as bordas de cada região que compõem cada objeto;
- cálculo do centro de massa, baseado no mínimo retângulo envolvente de cada objeto e de cada região;
- identificação das relações espaciais entre regiões vizinhas; e
- obtenção da informação de ângulo e do vetor normal de cada ponto de interseção entre as regiões;

Um exemplo deste processo é apresentado na Figura 28 a seguir. Na Figura 29a, é apresentada uma amostra de uma imagem range e na Figura 28b, é apresentado o conjunto das regiões que compõem a imagem da Figura 28a. Um esquema mais detalhado desta fase pode ser visto em [Vieira, 2002] .



(a) (b)

Figura 28: Exemplo de imagem range e a poligonização do objeto e de suas regiões.

Conforme visto no Capítulo 3.4, existem alguns complicadores na definição de um banco de dados para imagens. Várias pesquisas realizadas [Gaede & Gunter, 1998], apontam que os sistemas criados são bastante dependentes do domínio da aplicação, e são utilizadas características diferentes para recuperar as imagens. Nesse sentido é necessário identificar as características que serão usadas para prover uma indexação e uma recuperação eficientes no SRIC3D.

Como não estão definidas as características que podem ser usadas para prover uma recuperação eficiente de imagens de profundidade, a elaboração da estrutura para o armazenamento e indexação destes dados tem que ser extremamente flexível e adaptável, permitindo que sejam: inseridas novas imagens e características ao sistema; criadas outras estruturas de dados capazes de armazenar as novas características; realizados testes, com estas características fazendo parte da estrutura de indexação, de forma que este processo não inviabilize o desempenho geral do sistema e nem ocorra a perda das estruturas já criadas.

Inicialmente, tomando como base as características extraídas até o momento, é utilizado um módulo de mineração de dados descrito em [Vieira, 2002] para auxiliar na identificação das características que mais se destacam para classificar as imagens de profundidade. Este módulo realiza um estudo estatístico do comportamento das características das imagens armazenadas no banco de dados, elaborando regras para a classificação das imagens. Uma vez identificadas as principais características, as imagens são classificadas de acordo com a probabilidade de pertencerem ou não a uma das classes encontradas. O mesmo método é aplicado no momento de uma consulta, onde a imagem exemplo recebe também sua classificação e o sistema busca por imagens que possuam uma classificação similar.

Este processo servirá então para definir os atributos a serem utilizados como elementos principais do vetor de características, usado para formar o índice da tabela de imagens. É necessário definir também o método de acesso adequado para indexar os dados. O método de acesso utiliza um conjunto de operadores relacionais para

montar uma estrutura de indexação e preencher esta estrutura com os dados de referência. Este mesmo conjunto de operadores é utilizado para pesquisar na estrutura no momento de uma consulta.

Conforme visto no Capítulo 3, não existe um conjunto de operadores padrão para objetos multidimensionais [Gaede & Gunther, 1998] e, geralmente, também são dependentes do domínio da aplicação. Desta forma, o projeto exigirá a implementação de um certo número de operadores que suportem os tipos de dados e os tipos de consultas que serão inferidas ao sistema, mas também permitirá que novos operadores sejam cadastrados e utilizados na recuperação de imagens.

Baseado no exposto até o momento, os requisitos para a avaliação prática visando ao projeto de uma estrutura de indexação para imagens 3D englobam as seguintes atividades:

- Criação dos tipos de dados a serem usados para representar as imagens;
- Modelagem e criação de uma estrutura flexível para armazenar as informações sobre as características das imagens;
- Criação de operadores geométricos e relacionais para dar apoio à estrutura de indexação;
- Criação das funções que serão responsáveis pela montagem e manutenção da estrutura de indexação;
- Criação das funções que suportam a manipulação dos dados;

Para prover estes recursos é necessária a utilização de um banco de dados objeto-relacional, que permita a utilização tipos de dados definidos pelo usuário (UDT – *user data types*) e implemente o conceito de herança, características fundamentais para a criação de uma estrutura adaptável. Assim, será utilizado o banco de dados Postgres[8] como repositório dos dados do SRIC3D. Outra característica importante que

[8] http://www.postgresql.org

levou a escolha do Postgres foi o fato de possuir suporte a dados espaciais e permitir a criação de métodos de acesso espaciais e multidimensionais (por exemplo R-Trees).

6.1 – REPRESENTAÇÃO DAS IMAGENS RANGE

Para cadastrar estas informações na base de dados, é necessária uma modelagem adequada para garantir o armazenamento, a indexação e sua recuperação. Desta forma, teremos três categorias diferentes de atributos para armazenar as informações das imagens: atributos textuais, atributos visuais e atributos espaciais.

Para o conjunto de características extraídas até o momento, é utilizada a estrutura descrita a seguir.

6.1.1 – Representação por atributos Textuais

Os atributos textuais servem para registrar as informações de contexto da imagem: nome, origem, tipo do objeto, data de aquisição, etc e são inseridos manualmente no momento do cadastramento da imagem. Conforme pode ser visto na Tabela 2 a seguir:

Nome Físico	Descrição	Tipo/Tamanho (bytes)
Id_Object	Código de Identificação do objeto	*Serial (int4)*
Desc_Object	Nome do Objeto	*Varchar (40)*
Dt_Aquisition	Data de aquisição	*Date/time*
Id_Origin	Identificação da origem da imagem	*Int4*
Desc_Origin	Descrição da origem da imagem	*Varchar(60)*
Name_Resp	Nome do responsável pela captura da imagem	*Varchar(60)*
Id_Image	Identificação da imagem (vista) capturada	*Serial*
Raw_Image	O arquivo físico da imagem	*Blob*
Original_file	O arquivo de dados 3D da imagem	*Blob*
Angle	Ângulo da vista do objeto	*Int 4*

Tabela 2: Atributos textuais

6.1.2 – Representação por atributos Visuais

Na categoria das características visuais estão sendo extraídas, atualmente, as características: Forma (*shape*) através das informações de momentos (*moments*) e de textura através da matriz de co-ocorrência. Este mesmo conjunto também é gerado para cada região identificada em uma imagem. Desta forma teremos os seguintes atributos, descritos na Tabela 3:

nome físico	Descrição	Tipo/Tamanho (bytes)
angsecmoment	Momento angular secundário	Float
contrast	Contraste	Float
correlation	Correlação	Float
Variance	Variância	Float
invdiffmoment	Momento inverso diferencial	Float
sumaverage	Soma das médias	Float
sumvariance	Soma das variâncias	Float
sumentropy	Soma da entropia	Float
Entropy	Entropia	Float
diffvariance	Diferença de variância	Float
diffentropy	Diferença de entropia	Float
Meascorrelat1	Média correlata 1	Float
Meascorrelat2	Média correlata 2	Float
maxcorrelatcoef	Máximo Coeficiente de correlação	Float
mu00	Momentos Centrais	Float
mu11		Float
mu20		Float
mu02		Float
mu30		Float
mu03		Float
mu12		Float
mu21		Float
nu00	Momentos Centrais Normalizados	Float
nu11		Float
nu20		Float
nu02		Float
nu30		Float
nu03		Float
nu12		Float
nu21		Float
M1	Momentos Invariantes	Float
M2		Float
M3		Float
M4		Float
M5		Float
M6		Float
M7		Float

m00	Momentos Padrões	Float
m10		Float
m01		Float
m11		Float
m20		Float
m02		Float
m30		Float
m03		Float
m12		Float
m21		Float

Tabela 3: Atributos visuais.

6.1.3 – Representação por atributos Espaciais

A extração das informações espaciais caracteriza-se pela identificação de regiões de mesmas características no objeto, e pela identificação das relações espaciais entre as regiões identificadas. Os atributos estão especificados na Tabela 4 abaixo:

Nome físico	Descrição	Tipo/Tamanho (bytes)
bound_poly	Relação de pontos que formam o contorno do objeto	Polygon
center_point	Centro de massa do objeto ou da região	Float
Neighborhood	Relação de regiões vizinhas	Polygon
numberofvertices	Número de vértices	INTEGER
Numberofregions	Número de regiões	INTEGER
objectratio		Float
Área	Área do objeto da imagem	Float
Centerx	Coordenada x do centro de massa	Float
Centery	Coordenada y do centro de massa	Float
Centerz	Coordenada z do centro de massa	Float
Theta		Float
eccentricity		Float
uppercornerx	Canto superior x	Float
uppercornery	Canto superior y	Float
uppercornerz	Canto superior z	Float
bottomcornerx	Canto inferior x	Float
bottomcornery	Canto inferior y	Float
Bottomcornerz	Canto inferior z	Float
upperpoint	Ponto superior	Float
lowestpoint	Ponto inferior	Float
Leftpoint	Ponto mais a esquerda	Float
Rightpoint	Ponto mais a esquerda	Float

Tabela 4: Atributos Espaciais.

6.1.4 – Modelo de Dados Sugerido

Após a identificação das características extraídas, um modelo de dados inicial está apresentado na Figura 29 a seguir. Ele descreve a estrutura implementada e a forma com que estão relacionadas às características de uma imagem com seus atributos de identificação. Caso queira-se incluir mais um conjunto de características, basta-se acrescentar esta nova entidade ao modelo e ligar à entidade principal (*IMAGE*), através dos atributos de identificação.

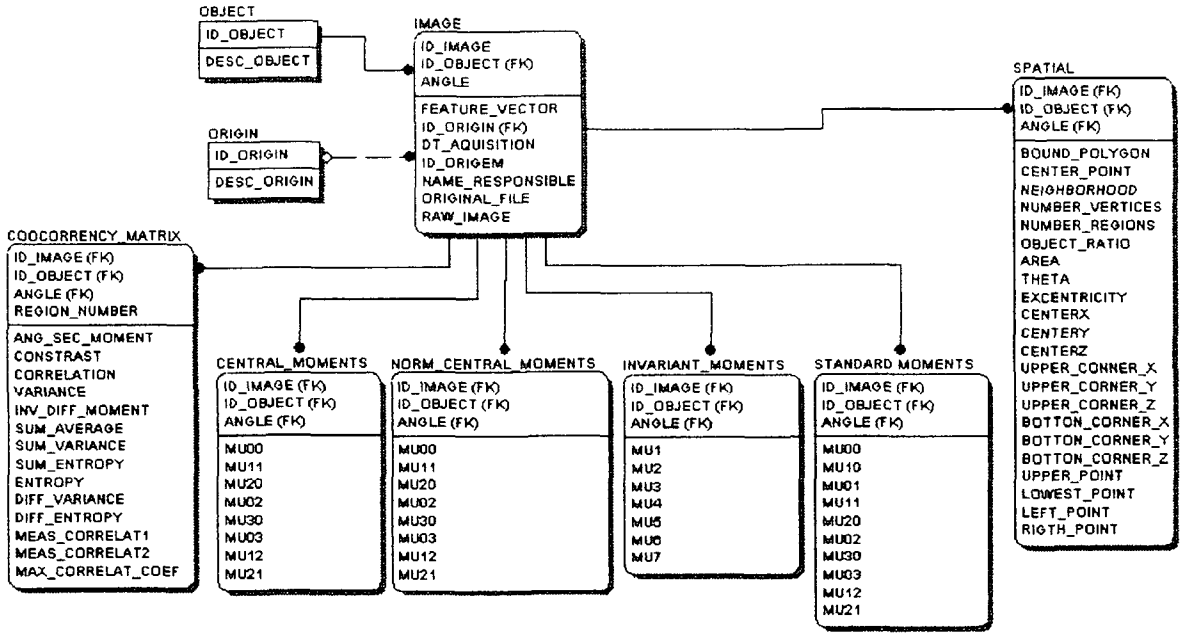


Figura 29: Modelo de Dadosdo SRIC3D..

6.2 – TIPOS DE DADOS PARA A REPRESENTAÇÃO DAS IMAGENS

Conforme visto no Capítulo 2, é bastante comum a utilização de um vetor de características para armazenar um conjunto de informações de uma imagem. Isto se justifica pela facilidade de manipulação dos dados provida pela estrutura do tipo vetor. No SRIC3D será adotada a mesma estrutura pois ainda não estão definidas quantas e nem quais as melhores características a serem utilizadas, ou que se adaptam melhor, a representação ou indexação de imagens de profundidade.

O Postgres possui diversos tipos de dados (listados no Anexo I) e também suporta vetores de alguns destes tipos. Seria possível aproveitar um deles para implementar os vetores de características, porém, optou-se pela definição de um tipo de dado que servirá de base para a criação de todos os vetores existentes no sistema, o tipo *vector*, que corresponde a um vetor n-dimensional do tipo *float8*.

Desta forma, caso observe-se que existe uma sub-utilização da estrutura ou um possível rompimento de um limite (*overflow*) é possível alterar apenas o tipo de dados básico, automaticamente modificando toda a estrutura, em vez de ter que trocar em todas as entidades criadas.

Para os tipos de dados e operações espaciais o Postgres está preparado para suportar dados espaciais bidimensionais, mas é interessante possibilitar que o sistema possa operar com pontos *n*-dimensionais, assim, foi criado tipo de dado chamado *n_point* que pode ser utilizado para representá-los.

Além de criar tipos de dados para representar vetores de características e pontos *n-dimensionais* são necessárias implementações de operadores relacionais e funções para auxiliar a manipulação e às operações destes tipos de dados.

6.3 – OPERADORES RELACIONAIS GEOMÉTRICOS

O Postgres já possui alguns operadores implementados, listados no Anexo II. Estes operadores são bastante versáteis para a operação com os tipos de dados espaciais já existentes, bem como, podem ser usados para operações de busca sobre a árvore de índice. Atualmente, existe apenas o operador de igualdade ("=") capaz de realizar alguma avaliação sobre os tipos de dados *vector* ou *n_point*. Este operador é capaz de receber dois pontos *n*-dimensionais ou dois valores do tipo *vector* e verificar se os valores de cada uma das dimensões são iguais. Em caso afirmativo retorna a valor TRUE.

Porém em alguns casos, este conjunto de operadores precisa ser adaptado para realizar as operações com os tipos de dados criados. Desta forma, foram criados os seguintes operadores:

6.3.1 – Operadores < e <=

Estes operadores recebem dois valores do tipo *vector* ou *n_point* e verificam se todos os valores de todas as dimensões de um vetor são menores, ou iguais para o segundo caso, que seus correspondentes no outro vetor, retornando TRUE em caso afirmativo. Em outras palavras, caso uma das dimensões do primeiro seja MAIOR que seu correspondente no segundo, o retorno será: FALSE.

6.3.2 – Operador > e >=

Estes operadores recebem dois valores do tipo *vector* ou *n_point* e verificam se todos os valores de todas as dimensões de um vetor são maiores, ou iguais no segundo caso, que seus correspondentes no outro vetor, retornando TRUE. Em outras palavras, caso uma das dimensões do primeiro seja MENOR que seu correspondente no segundo, o retorno será: FALSE.

6.3.3 – Operador ~=

Este operador recebe dois valores do tipo *vector* ou *n_point* e verifica se o valor do segundo vetor é similar ao primeiro. Esta similaridade é definida por um conjunto de tabelas que permitem a configuração de um valor aceitável para a diferença de valores entre os vetores, conforme mostra o Capítulo 6.5.

6.4 – FUNÇÕES DE APOIO PARA OS OPERADORES

Como o Postgres é um banco de dados objeto-relacional, cada entidade é tratada como uma classe distinta. Dessa forma, possui seus atributos, seus métodos e seus operadores. Para que seja possível manipular qualquer atributo de uma classe, é necessário utilizar um de seus métodos, que são, neste caso, funções armazenadas no banco de dados. Assim, para cada novo operador criado existe a respectiva função que executa a operação.

Da mesma forma, para manipular os dados, através de comandos SQL, é necessário que existam funções capazes de realizar esta tarefa. O mesmo se aplica às operações de manutenção da estrutura de indexação e a manutenção de tabelas de sistema. Foram criadas as seguintes instruções para manipular os tipos de dados criados:

- *Length* : retorna o tamanho do vetor, ou seja o seu número de dimensões;
- *great* e *greateq* : implementam os operadores > e >= respectivamente;
- *less* e *lesseq*: implementam os operadores < e <= respectivamente;
- *similar*: implementa o operador ~=;

6.5 – ESTRUTURA DE INDEXAÇÃO

Normalmente, em sistemas de bancos de dados convencionais, as informações cadastradas são armazenadas em estruturas chamadas de *páginas de dados* as quais possuem um tamanho fixo pré-definido. Quando um índice é criado sobre os atributos de uma tabela, é criada uma estrutura, como por exemplo uma árvore B, onde as folhas da árvore apontam para a página de dados onde se encontra a informação procurada.

Para o SRIC3D a estrutura de indexação deve adaptar-se aos critérios de classificação propostos pelos algoritmos de mineração de dados e indexar as imagens

pelas características (*features*) propostas pelo módulo. Para permitir tal flexibilidade ao longo do projeto, optou-se pela criação de uma outra entidade (tabela) formada pela união dos atributos de identificação da imagem (*id_image* e *id_object* ou outros que possam surgir) e um atributo que contém o vetor de características das imagens. A entidade criada, *indexedfeature*, possui a estrutura conforme definida na Tabela 5, a seguir:

Nome Físico	Descrição	Tipo/Tamanho (bytes)
id_image	Identificador da imagem.	`Integer`
id_object	Identificador do objeto	`Integer`
feature_vector	Vetor de características.	`Vector ou n point`

Tabela 5: Estrutura da tabela indexada.

No processo implementado, os atributos de uma ou mais tabelas distintas têm seus valores cadastrados no vetor de características da entidade de dados de indexação. Ou seja, os valores dos atributos que serão usados para indexar a base de dados, são copiados para um vetor na nova tabela e o índice é criado sobre este vetor.

Desta forma, é possível que qualquer atributo de qualquer tabela faça parte da chave de pesquisa, permitindo ao sistema adaptar-se às características definidas pelo módulo de mineração e também permitindo que o usuário defina por quais características ele pretende indexar as imagens da base, criando uma maneira simples de avaliar outras características das imagens sem prejudicar a estrutura atual. É necessário, então, que o módulo de mineração, ou o usuário, defina quais são os atributos a serem usados para a indexação e qual o método de acesso escolhido.

Definidas as características, é preciso alimentar os dados na tabela onde os dados serão indexados. Para evitar a necessidade da digitação de comandos para a inclusão dos dados na estrutura, foi criada uma tabela auxiliar identificada como *featurelist*, descrita na Tabela 6, onde devem ser registrados os nomes das tabelas e dos atributos que servirão como índice.

Nome Físico	Descrição	Tipo/Tamanho (bytes)
Tbname	Nome da entidade.	VARCHAR(60)
Colname	Nome do campo a ser usado para a indexação	VARCHAR(60)
accessmethod	Método de acesso (0 - R-Tree ou 1 - M-Tree)	INTEGER
id_strategy	Identificação da estratégia de afrouxamento	INTEGER
Codstatus	Situação : 1 - Ativo , 0 - Inativo	INTEGER

Tabela 6: Estrutura de cadastramento de atributos a serem usados pelo índice.

Assim que as informações sobre os atributos forem inseridas nesta tabela, é possível carregar a tabela de características indexadas (*indexedfeature*) com os valores dos atributos escolhidos. Esta carga é feita por meio da chamada de uma *stored procedure*[9], cada vez que se desejar criar uma estrutura de indexação para avaliação. Da mesma forma, durante o cadastramento de novas imagens na estrutura existente, *triggers*[10] garantem a inclusão de novos dados na tabela que contém os dados a serem indexados.

6.5.1 – Método de Acesso

O Postgres possui algumas funções de suporte e operadores implementados para manipular estruturas de indexação sobre tipos de dados espaciais. Porém os tipos são apenas três: BOX ,BIGBOX e POLYGON , sendo que as operações suportadas podem ser encontradas no Anexo III.

Em geral estas operações podem ser: *sobrepõe (overlap), sobrepõe à esquerda, sobrepõe à direita, está à esquerda, está à direita, está contido, contém (outro polígono ou ponto)*, entre outras. Porém sua utilização está limitada a representação da imagem (ou região) por um polígono (seqüência de pontos no espaço) ou por um retângulo (MBR). Para utilizar este conjunto de operadores e métodos de acesso, foi preciso entender como eles são implementados no Postgres e adaptar esta estrutura aos tipos de dados criados para o SRIC3D.

A maior dificuldade em construir um método de acesso é a implementação de uma função de similaridade, pois ela é usada em pesquisas do tipo *nearest_neighbor*. Como não se tem ainda muito conhecimento das possíveis relações existentes entre as características, o objetivo deste trabalho é justamente elaborar uma

---

[9] Conjunto de rotinas que fica armazenado internamente no banco.

[10] Programas que são executados quando um evento de alteração de dados ocorre: INSERT, UPDATE ou DELETE.

estrutura flexível que permita a realização de pesquisas na busca por uma função de similaridade entre as características das imagens *range*.

6.6 – FUNÇÃO DE SIMILARIDADE

Como [Petrakis, 2002a] sugere, as imagens não são idênticas, tornando-se necessária uma função que determine a similaridade entre imagens. Desta forma, é preciso implementar uma forma de determinar a similaridade entre imagens *range* do mesmo objeto.

O módulo de mineração de imagens identifica as características para a classificação do conjunto de dados, mas deve haver um operador que recupere estas informações. No Postgres os operadores são suportados por funções que recebem os dados e os transformam na resposta adequada. Os operadores relacionais fazem um processamento com os dados fornecidos e retornam *TRUE* ou *FALSE* de acordo com a situação. Se fosse usado um operador comum de igualdade ou equivalência para comparar duas imagens distintas de um mesmo objeto, muito provavelmente a função retornaria que as imagens são divergentes, pois uma pequena diferença em um único valor de uma das dimensões seria suficiente para invalidar a operação.

Assim, foi criado o operador "~=" com o significado de "*é similar*", que é implementado por uma função de similaridade. Este operador realiza um teste lógico verificando se os valores das características fornecidas diferem em uma pequena porcentagem ou se esta diferença está dentro de um valor tolerável. Desta forma este operador, baseado em uma função de similaridade, pode determinar se uma imagem é similar à outra.

Para permitir ajustes nesta função que implementa o operador *é similar* foi definido o conceito de "estratégia" (*strategy*). A estratégia permite a configuração da função de similaridade de acordo com alguns parâmetros cadastrados em uma tabela do banco de dados que podem ser alterados a qualquer momento. Os dados são cadastrados de acordo com a estratégia a ser utilizada para relaxar a busca de

elementos pela árvore. Pode-se configurar o coeficiente de afrouxamento para cada uma das dimensões do vetor individualmente ou um valor geral para todas as dimensões (*id_dimension* = 0). A tabela do banco de dados onde os parâmetros são inseridos é a *strategy*, descrita a seguir:

Nome Físico	Descrição	Tipo/Tamanho (bytes)
cod_strategy	Identifica a estratégia a ser adotada	Integer
id_dimension	Identifica a dimensão (0 para todas)	Integer
coef_relaxation	Valor percentual de relaxamento para a dimensão	Float
desc_strategy	Descritivo sobre a estratégia implementada	Varchar(50)

Tabela 8: Estrutura da entidade strategy.

Como ainda não foi definida uma forma correta de determinar a similaridade entre imagens *range*, então, optou-se por implementar similaridade utilizando algumas estratégias, tais como a distância euclideana, realizando alguns testes para avaliar o desempenho da estrutura criada no banco de dados. Estes testes estão descritos no Capítulo 7.

6.7 – A CRIAÇÃO DE UM ÍNDICE

Estando os componentes principais criados no banco de dados, o processo de criação de uma estrutura indexada para um conjunto de características é bem simples. Inicialmente é preciso identificar a tabela, seus atributos e o método de acesso para a indexação. Isso é realizado através da inclusão destas informações na tabela *featurelist*.

Tomando como exemplo a tabela *baseimage* atualmente criada e carregada com informações de um conjunto de imagens, vamos definir uma indexação sobre alguns de seus atributos. Por exemplo: *objectratio, theta e eccentricity*. O método de acesso utilizado é a *R-Tree*. Com isso, a tabela *featurelist* ficou conforme apresentado na Tabela 7:

Tbname	Colname	Access_Method	Cod_situacao
baseimage	Objectratio	1	1
baseimage	Theta	1	1
baseimage	Eccentricity	1	1

Tabela 7: Conteúdo da tabela de configuração.

Para se ter idéia da flexibilidade proporcionada por esta estrutura, é possível utilizar várias estratégias diferentes para um conjunto diferente de dados. A Tabela 8 abaixo, mostra alguns exemplos das estratégias utilizadas na realização dos testes apresentados no Capítulo 7.

Cod_strategy	Id_dimension	Coef_relaxation	Desc_Strategy
0	0	0,0005	Variação de 0.005% na Distância Euclideana entre vetores
1	0	0,012	Variação de 1,2% na Distância entre vetores
2	1	0,1	Variação de 10% no valor para esta dimensão
2	2	0,015	Variação de 1.5% no valor para esta dimensão
2	3	0,00015	Variação de 0.0015% no valor para esta dimensão
3	1	0.025	Valor fixo para a dimensão 1
3	2	0.05	Valor fixo para a dimensão 2
3	3	1	Valor fixo para a dimensão 3.

Tabela 8: Relação das estratégias.

A primeira estratégia (cod_strategy = 0) utiliza-se da distância euclideana para determinar a distância entre os pontos. Porém, ao se obter o valor calculado pela função, ainda não é possível saber se os pontos estão distantes ou próximos um do outro. A função então determina a distância entre um dos pontos e o ponto zero do espaço multidimensional. Assim, é possível comparar se o valor calculado da distância euclideana entre os pontos é um valor razoável em relação à distância entre um dos pontos e o ponto zero. A faixa de tolerância é, então, determinada pelo coeficiente de relaxação (*coef_relaxation*). O algoritmo implementado na função de similaridade é:

```
/* Distância Euclideana simples */B
retval := TRUE;
FOR i IN 1..length(pontoA) loop
      distancia := distancia + (pontoA[i] - pontoB[i])**2;
      precision := precision + (pontoA[i] **2 );
end loop;
distancia := sqrt(distancia);
precision := sqrt(precision)* coef_relaxation;
if distancia > precision then
   retval := FALSE;
return retval;
```

Entretanto esta estratégia não é recomendada quando as características manipuladas possuam valores de grandezas muito distintas, pois uma pequena variação na dimensão de maior variância, poderia eliminar uma boa candidata.

Já a estratégia 1, utiliza um percentual do valor de cada dimensão para determinar se a diferença entre os valores de cada dimensão é tolerável. Ou seja, a diferença entre cada valor deve ser menor que o percentual definido pela estratégia. No exemplo apresentado na Tabela 8, caso a diferença entre os valores de uma determinada dimensão seja menor que 1,2% do valor de um dos pontos para aquela dimensão, então eles são considerados similares.

Nesta estratégia, o número de operações de comparação a ser realizada depende exclusivamente dos dados, o que dificulta a definição da ordem de complexidade ou a estimativa de custo de execução de uma pesquisa com este tipo de estratégia, conforme pode ser observado no algoritmo abaixo:

```
/* Distância estimada por % do valor da dimensão */
retval := TRUE;
FOR i IN 1..length(pontoA) LOOP
    distancia := abs(pontoA[i] - pontoB[i]);
    precision := pontoA[i] * coef_relaxation[i];
    if distancia >= precision then
       retval := FALSE;
       i:= length(pontoA);
    end if;
end LOOP;
return retval;
```

A estratégia 2 é uma flexibilização da estratégia 1. O princípio é o mesmo, um valor percentual do valor da dimensão é aceito como diferença entre os valores dos pontos para aquela dimensão. A vantagem, neste caso, é que se pode colocar valores percentuais diferentes a cada dimensão.

A estratégia 3, também derivada da estratégia 1, permite a utilização de um valor fixo (discreto) para cada dimensão e não mais proporcional ao valor existente em uma das dimensões. Neste caso, o coeficiente de relaxação é um valor distinto para cada dimensão, conforme o algoritmo a seguir:

```
/* Distância estimada por valor discreto por dimensão */
retval := TRUE;
FOR i IN 1..length(pontoA) LOOP
    distancia := pontoA[i] - pontoB[i];
    if distancia >= coef_relaxation[i] then
```

```
        i:= length(pontoA);
        retval := FALSE;
      else
         i:= i+1;
    end if;
end LOOP;
return retval;
```

As estratégias implementadas são sugestões de funções para determinar a similaridade entre objetos. O objetivo neste trabalho não é determinar a melhor estratégia para determinar similaridade entre imagens range, e sim disponibilizar um ambiente para pesquisas e auxiliar o processo de recuperação destas imagens. Assim que seja possível determinar uma função de similaridade entre as imagens *range* através de suas características pode-se alterar ou modificar as estratégias atuais por meio da substituição da função *similar*.

6.8 – O CUSTO DE UMA BUSCA

Um dos fatores que torna mais eficiente a busca pela informação em uma estrutura de indexação em vez de procurá-la diretamente na tabela onde está ela armazenada, é que a quantidade de leituras a ser realizada, em uma estrutura de indexação, é bem menor que a quantidade de leituras a ser realizada por uma leitura seqüencial de toda uma tabela. Isso envolve não só operações de I/O como também alocação de dados na memória do SGBD (*cache*).

No caso específico do Postgres, uma página de dados possui tamanho de 8192 bytes. Para cada linha existente em uma tabela, são usados 48 bytes para informações de controle. Supondo que, por exemplo, cada registro tenha um tamanho 148 bytes em cada página de dados é possível armazenar aproximadamente 41 registros.

Quando se cria um índice, as informações contidas na árvore de índice também são armazenadas em uma página, denominada *página de índice*. Esta página também tem o mesmo tamanho de uma página de dados, porém as informações a

serem armazenadas são informações sobre a estrutura de índice que depende dos atributos usados pela estrutura. Por exemplo, tomemos uma árvore do tipo R-Tree.

Conforme visto na Capítulo 4.1.2.8, uma R-Tree armazena em seus nós *não-folha* informações sobre os limites inferior e superior da MBR coberta por aquele nó e ponteiros para os nós que pertencem àquela região. Caso seja um *nó-folha*, as informações armazenadas são ponteiros para as páginas da tabela onde os objetos (pontos ou polígonos) que fazem parte daquela MBR. Assim, uma estrutura R-Tree simples definida por duas coordenadas de pontos X1Y1 e X2Y2, do tipo *point* (*float4*) ocupam 16 bytes da página de índice, mais um inteiro de quatro bytes (*int4*) para cada um de seus *nós-filhos*. Cada um dos *nós-folhas* precisa de 48 bytes para apontar para uma página de dados. A quantidade de páginas de índices necessárias para indexar uma tabela pode ser estimada pelas equações a seguir:

- Número de páginas para nó folha:

((float4 * num. dimensões) + 48 + OID)* número de registros da tabela) / 8192)

- Número de páginas para nós não-folha:

((2* float4 * num. dimensões)+ 48+ OID) * núm. de páginas de nós-folhas) /8192

Exemplo:

Supondo que seja utilizado um vetor de características de 4 (quatro) dimensões para indexar uma tabela com 10.000 registros, e cada registro da tabela tenha 148 bytes de tamanho. Os valores seriam os seguintes:

- Num. páginas de dados: ((148+48)*10000)/8192 = 239 páginas.
- Núm. páginas de nós-folhas: ((4*4)+48+4)* 10000) /8192 = 83 páginas
- Num. páginas de nós não folha: ((2*4*4)+48+4)* 83)/8192 = 1 página.

Assim, uma página de índices para uma R-Tree é capaz de indexar 10000 registros em 83 páginas de dados. Não podemos esquecer que em uma R-Tree existe a sobreposição de regiões, isto significa que para recuperar informações a partir de uma consulta, neste caso específico, são necessárias: uma leitura na página de índices de nós não-folhas identificando os nós da árvore a serem percorridos, mais uma leitura

para cada nó folha percorrido, mais uma leitura da página de dados onde se encontra o registro especificado pelos valores encontrados no nós-folhas.

Com certeza o custo computacional para ler esta estrutura é bem menor que uma leitura completa da tabela. Porém deve-se observar que, caso a faixa de valores torne-se muito grande, o otimizador de consultas do próprio gerenciador do banco de dados pode decidir ler a tabela inteira.

Para evitar isso, além das estatísticas que devem ser colhidas para servir de base para o algoritmo de decisão do plano de execução[11], existe uma função utilizada pelo otimizador identificada pelo atributo *amcostestimate* de cada método de acesso. Esta função auxilia o algoritmo de decisão a calcular uma estimativa de custo de pesquisa através do método de acesso implementado pelo índice. Ver detalhes no Capítulo 7.1.

[11] Plano de execução é um algoritmo que determina de que forma o SGBD vai realizar a busca dos dados requisitados por uma consulta.

7. TESTES REALIZADOS

Conforme visto no Capítulo 3.3.3 existem algumas premissas que um método de acesso deve possuir para ser considerado eficiente: eficiência no tempo e no espaço, escalabilidade, dinamismo, etc. Assim, para avaliar a eficiência da estrutura criada, foi realizado um conjunto de testes visando avaliar se o índice é utilizado na recuperação das informações e se este acesso é mais rápido que uma busca seqüencial sobre uma tabela.

7.1 – CRITÉRIO PARA ESTIMATIVA

Todo SGBD possui um mecanismo interno, chamado otimizador de consultas (*query optimizer),* que identifica a melhor maneira de responder a uma consulta com o menor esforço, leia-se: menor utilização de recursos (memória, CPU e I/O). O otimizador colhe informações sobre a quantidade de leituras às páginas de dados a serem realizadas para executar a consulta solicitada, e com base nestes dados, decide entre uma busca seqüencial nas páginas da tabela (*full table scan*) ou por uma busca nas páginas de índice (*index scan*).

Para determinar o *esforço* para executar a consulta solicitada, o otimizador do Postgres mede o custo de processamento de quatro operações básicas, que são comumente realizadas na busca por uma informação:

- Acesso seqüencial a uma tupla de índice (*CPU INDEX TUPLE COST*);
- Acesso seqüencial a uma tupla de uma tabela (*CPU TUPLE COST*);
- Processamento de um operador existente na clausula *WHERE* da consulta (*CPU OPERATOR COST*);
- Acesso randômico a uma página de dados (*RANDOM PAGE COST*);

A cada consulta submetida ao SGBD, o otimizador identifica quais são as operações a serem realizadas para executar a consulta e por meio do custo de cada uma

delas estima um valor chamado de "Custo SQL" (*SQL Cost*) que é utilizado para valorar a melhor alternativa para realizar a busca. As tarefas básicas monitoradas possuem um custo estimado e seus valores *default* estão relacionados na Tabela 9 a seguir.

Tarefa	Custo
random_page_cost	4
cpu_tuple_cost	0.01
cpu_index_tuple_cost	0.001
cpu_operator_cost	0.0025

Tabela 9: Custo de cada operação no Postgres

Estes valores indicam que para realizar uma busca por toda uma tabela procurando por um valor exato, é realizada uma operação de comparação (*operator_cpu_cost*) entre o valor procurado e cada registro (*cpu_tuple_cost*). Assim, o custo computado pelo Postgres para a pesquisa completa em uma tabela de 100 registros é de 2.25, onde o valor excedente (1.0) se deve a operação de leitura de uma página de dados. Este custo permanece constante a cada incremento de 100 registros.

Para visualizar esta informação a maioria dos SGBD fornece o plano de execução de uma consulta (*query plan*) que contém o custo SQL para a consulta solicitada. Assim, para avaliar o desempenho da estrutura, utilizou-se, então, o resultado provido pelo otimizador quando este é submetido a uma consulta.

7.2 – VERIFICAÇÃO DA UTILIZAÇÃO DO ÍNDICE

Um dos primeiros testes realizados foi uma verificação da real utilização do índice pelo SGBD. Para isso, foram realizadas inclusões de vários registros em uma tabela e verificado se o índice era utilizado em consultas por valor exato (*point-query*). O gráfico 1 mostra o resultado do teste, comparando o custo de uma busca sem índice (*full table scan*) e o custo onde os dados são indexados por uma R-tree.

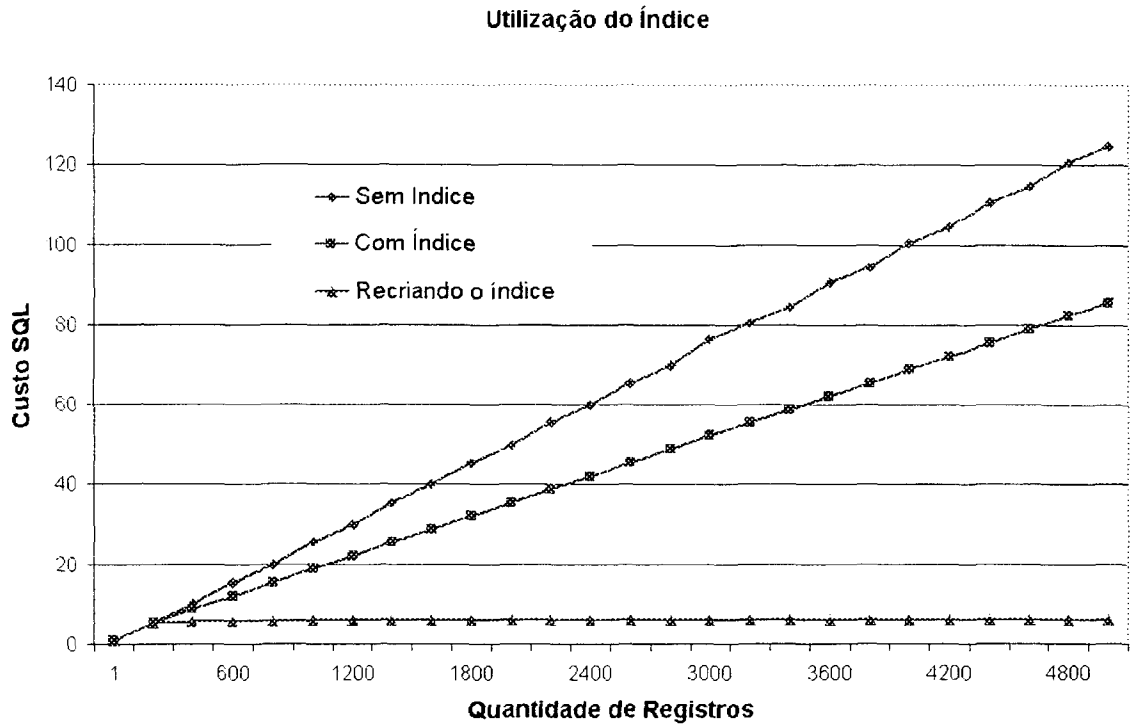


Gráfico 1: Busca por um valor exato sem índice e com índice R-Tree.

Com a realização do teste, observou-se que, para uma pequena quantidade de registros a serem pesquisados, o SGBD decide por fazer uma pesquisa direta na tabela (*full table scan*) não utilizando o índice, porém conforme o número de registros aumenta, o índice permite a recuperação da informação com um custo SQL menor, ou seja em um tempo menor.

Observou-se também que o controle estatístico do Postgres funciona apenas na criação de um índice, ou seja, quando se cria um índice o SGBD atualiza uma tabela com dados estatísticos sobre os valores do(s) atributo(s) utilizados como índice. Desta forma, não há uma atualização destes dados estatísticos a cada linha inserida ou a cada bloco inserido. Para verificar esta característica do SGBD, o teste acima foi realizado de duas formas diferentes. Na primeira o índice é criado com a tabela sem nenhum registro e os dados são inseridos em blocos de 200 e os valores de custo estatístico são colhidos. Na segunda, o índice é recriado a cada bloco de 200 registros e os dados estatísticos são colhidos. Percebe-se uma grande diferença no desempenho da estrutura sempre que o índice é criado.

7.3 – COMPARAÇÃO ENTRE B-TREE E R-TREE

Outro teste realizado foi uma comparação do desempenho da estrutura R-Tree criada em relação a uma estrutura B-Tree. Foram escolhidos alguns atributos e sobre eles foram criados índices utilizando o método de acesso da B-Tree e outro pela R-Tree. Os testes foram realizados para índices B-Tree de 1, 2 e 3 atributos e para a R-Tree comparou-se com vetores de 1, 2 e 3 dimensões respectivamente.

O teste consiste em popular uma tabela com uma quantidade de registros e avaliar o tempo e os recursos utilizados para a recuperação das informações aplicando as consultas por um único valor (*point-query*). O resultado do teste pode ser observado no Gráfico 2.

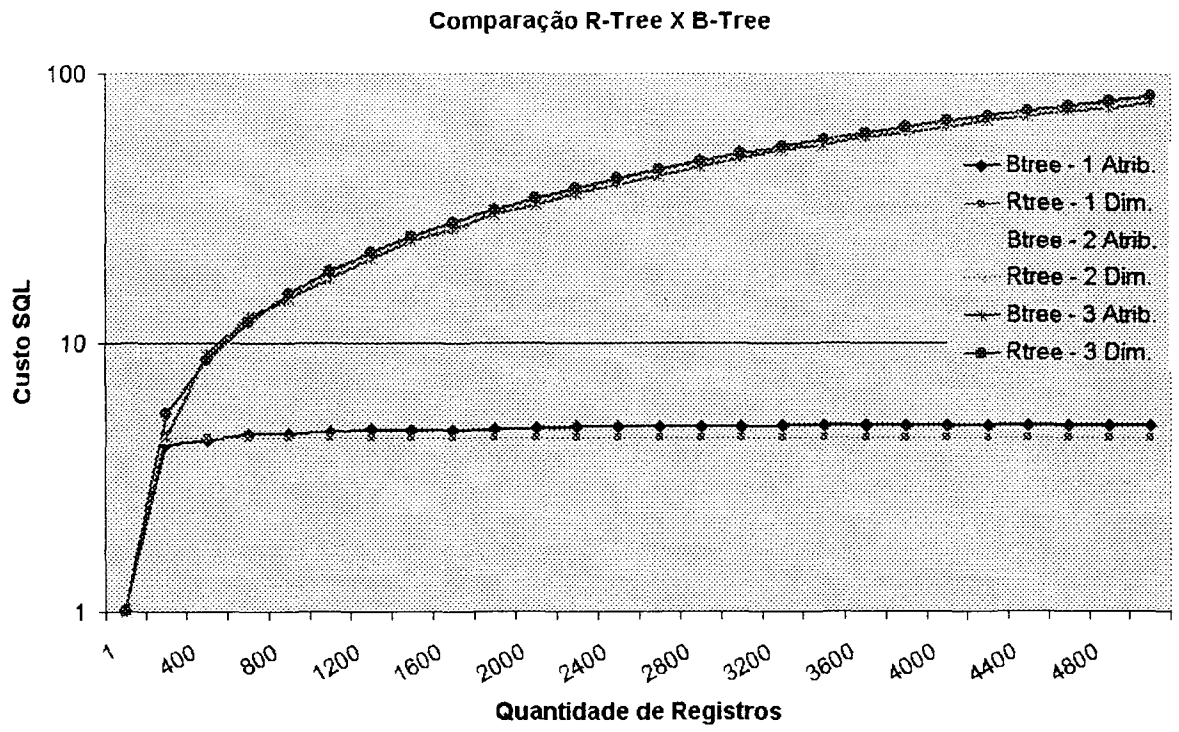
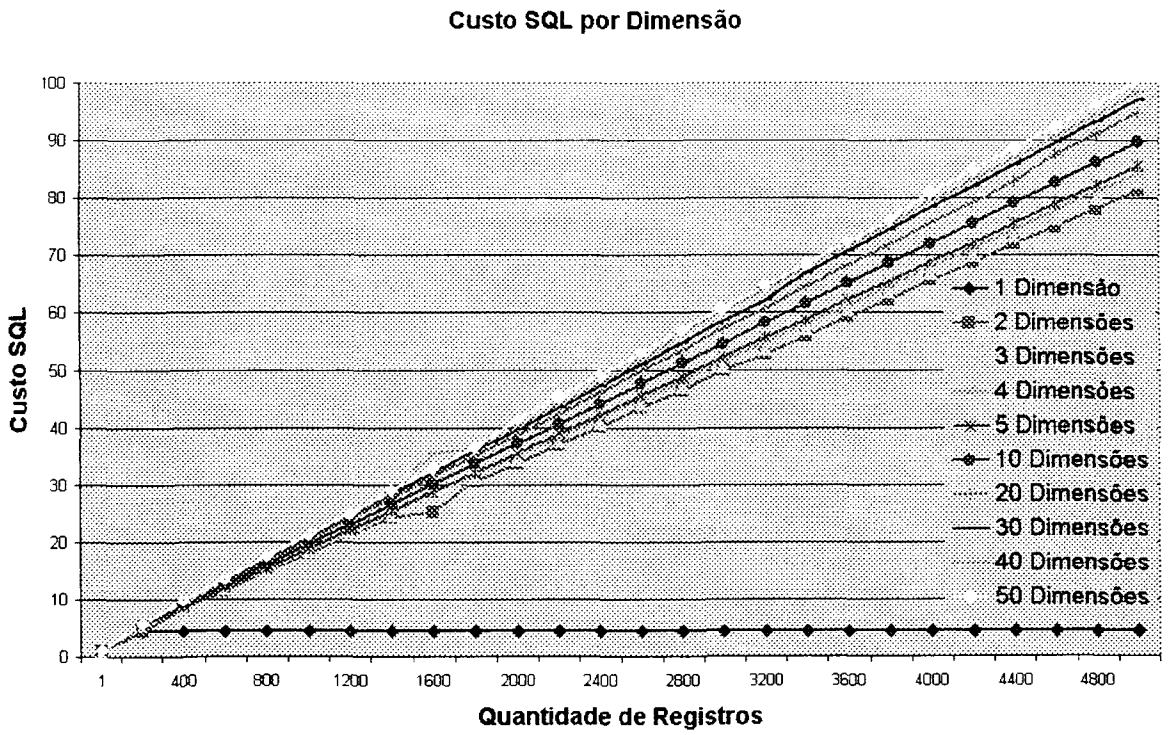


Gráfico 2: Comparação entre R-Tree e B-Tree.

Pode-se observar que o desempenho da B-Tree é um pouco melhor com valores pequenos, porém os custos aumentam conforme aumenta a quantidade de registros. Com base nestes dados foi possível comprovar que o desempenho da estrutura R-tree criada teve um desempenho similar ao da estrutura B-Tree.

7.4 – AVALIAÇÃO DO AUMENTO DAS DIMENSÕES

Outro teste realizado foi a verificação da influência da dimensão do vetor de características no desempenho da estrutura. Para isso, foram aplicadas consultas por valor exato com vetores de diferentes tamanhos e com quantidades de registros diferentes. O objetivo é avaliar o tempo de resposta, ou seja, o custo para o SGBD e seu impacto sobre o desempenho do banco. O Gráfico 3 apresenta o resultado deste teste. Nele pode-se verificar que a estrutura R-Tree implementada atende de forma satisfatória a uma quantidade de até 50 dimensões. Não foi possível avaliar o desempenho com um número maior de dimensões, pois até o momento não haviam mais características extraídas das imagens que pudessem ser usadas para compor o



vetor.

Gráfico 3: Avaliação do aumento das dimensões do vetor de características.

Comparando o resultado deste teste, apresentado no Gráfico 3, com o Gráfico 2, observa-se que o desempenho da estrutura mudou pouco com o aumento das dimensões. Pretende-se ainda aumentar a quantidade de dimensões para valores maiores que 100 para observar o desempenho.

7.5 – AVALIAÇÃO DA DISTRIBUIÇÃO DOS DADOS

Conforme [Gaede & Gunter, 1998], uma estrutura de indexação deve ser independente da distribuição dos dados de entrada. Para verificar isso, a estrutura criada foi submetida a uma série de consultas por valores exatos, procurando valores que estão em regiões distintas do espaço de dados. O gráfico 4 mostra a região do espaço multidimensional utilizado neste teste e os pontos escolhidos para serem usados na consulta. O desempenho do sistema, nas consultas formuladas, não varia com a distribuição dos dados ou com o ponto escolhido como elemento de pesquisa.

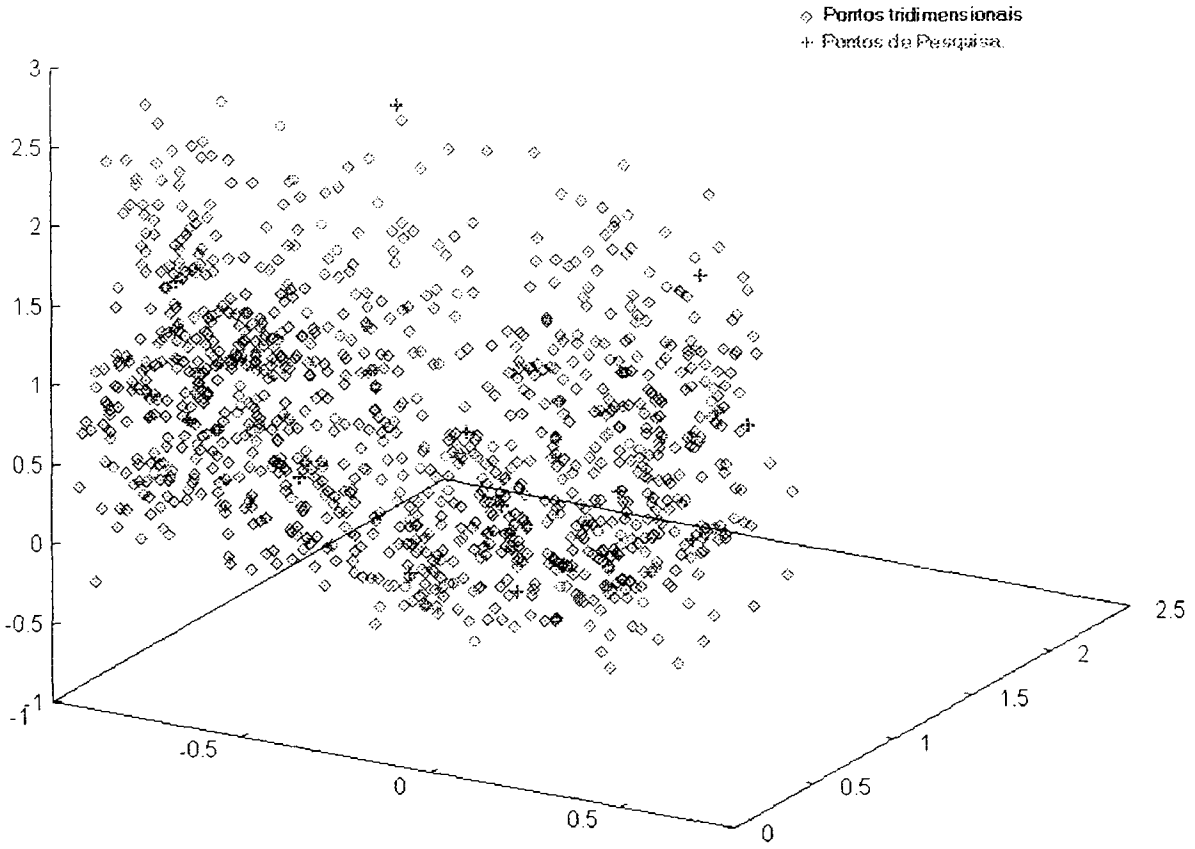


Gráfico 4: Distribuição dos dados no espaço testado.

7.6 – CONJUNTO DE OPERADORES

Ainda com relação aos itens apontados na seção 3.3.3, o número de operadores suportados pelo método de acesso deve ser considerado, pois apenas um

tipo de operador limita o tipo de consultas que podem ser inferidos ao sistema. Na estrutura atual, temos apenas 6 operadores para realizar operações com o novo tipo de dados criado.

Para avaliar estes operadores, foram submetidas consultas do tipo *point-query*, *range-query* e por valores similares a um determinado ponto (*nearest-neighbor-query*). O desempenho do sistema foi satisfatório nas consultas do tipo *point-query*, pois o operador de igualdade possui excelente seletividade ao realizar pesquisas na árvore de índices. Porém, o desempenho das consultas do tipo *range-query* não foi muito satisfatório. O otimizador do SGBD não consegue definir precisamente se é menos custoso realizar a pesquisa dos dados na árvore de índices ou diretamente na tabela, então decide, na maioria dos casos por fazer um *full table scan*, realizando a pesquisa utilizando o índice quando a faixa a ser pesquisada é bastante estreita.

A pesquisa por valores similares implementadas utilizando o conceito de estratégia demonstra uma das grandes facilidades providas por este trabalho. Conforme apresentado nas Seções 6.6 e 6.7, foi implementada uma estrutura especial para auxiliar na realização de consultas do tipo *nearest-neighbor*, porém vale lembrar que uma função de similaridade, neste caso, é fundamental para determinar a distância entre dois pontos. Como ainda não se obteve todas as relações possíveis entre características de imagens *range*, o objetivo é configurar critérios de aceitação de valores próximos ao valor utilizado como ponto de pesquisa, permitindo a identificação de uma função de similaridade ou de uma percentual de afrouxamento na busca que seja capaz de selecionar boas candidatas.

Foram escolhidos alguns pontos dentro do espaço multidimensional e destes pontos selecionou-se 5 para servirem como pontos de pesquisa. Estes pontos sempre aparecem em destaque nos gráficos. Desta forma, no momento da consulta o operador "similar", definido na Seção 6.3, verifica qual a estratégia a ser utilizada e avalia cada registro pesquisado conforme os critérios definidos.

O conjunto de gráficos a seguir apresenta algumas consultas realizadas seguindo os valores configurados na Seção 6.7, onde estão definidas 4 estratégias diferentes. O Gráfico 6a apresenta a estratégia 0, onde se buscam os valores com uma tolerância de 0.05% em todas as dimensões. Como foi usada a distância euclideana, e existe uma diferença na faixa de valores das dimensões utilizadas, o valor de tolerância é proporcional a maior das grandezas de valores das dimensões. Assim, conforme mostra o gráfico, um único ponto de pesquisa retorna quase todo o universo de dados do espaço de teste. Esta parece não ser uma boa estratégia quando existem características nos vetores de faixas muito diferentes. Ainda assim, o resultado obtido pode não refletir a realidade na relação entre características de imagens similares.

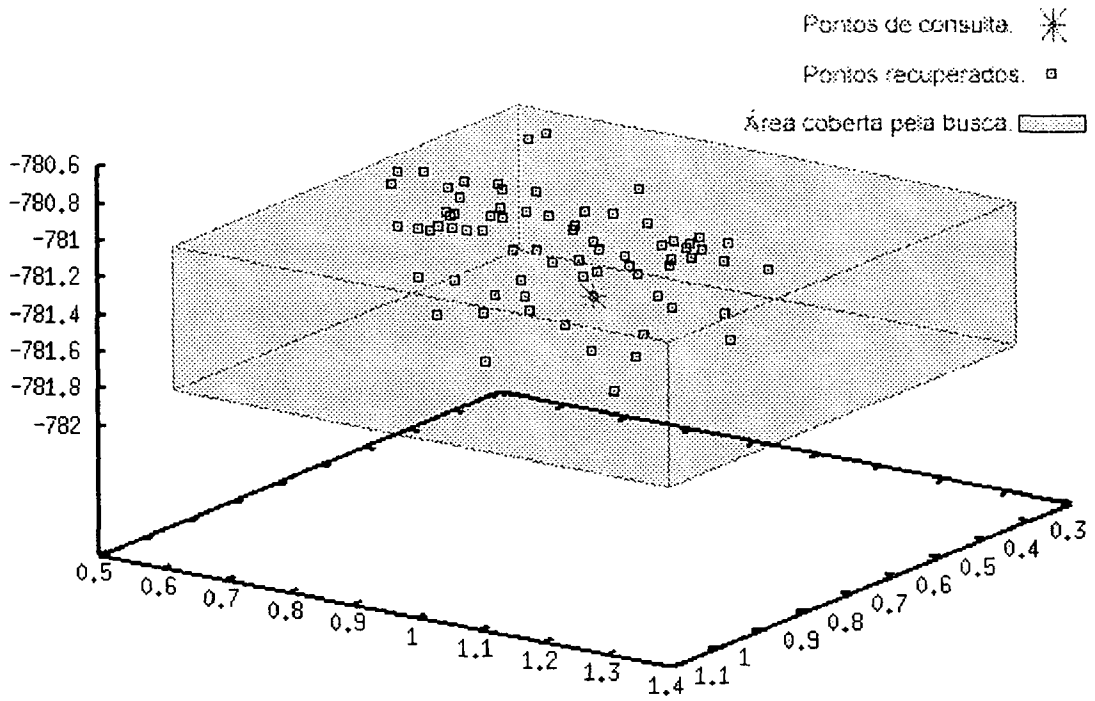


Gráfico 6a: Pesquisa por pontos similares usando a Estratégia 0.

O Gráfico 6b, apresenta o resultado da Estratégia 1, onde é a aceitação de um determinado ponto depende de um percentual do valor de cada dimensão de um dos elementos, ou seja, o valor de aceitação é proporcional ao valor da dimensão. Neste caso, utilizou-se um percentual de 1.2% da escala, o que resultou no gráfico a seguir.

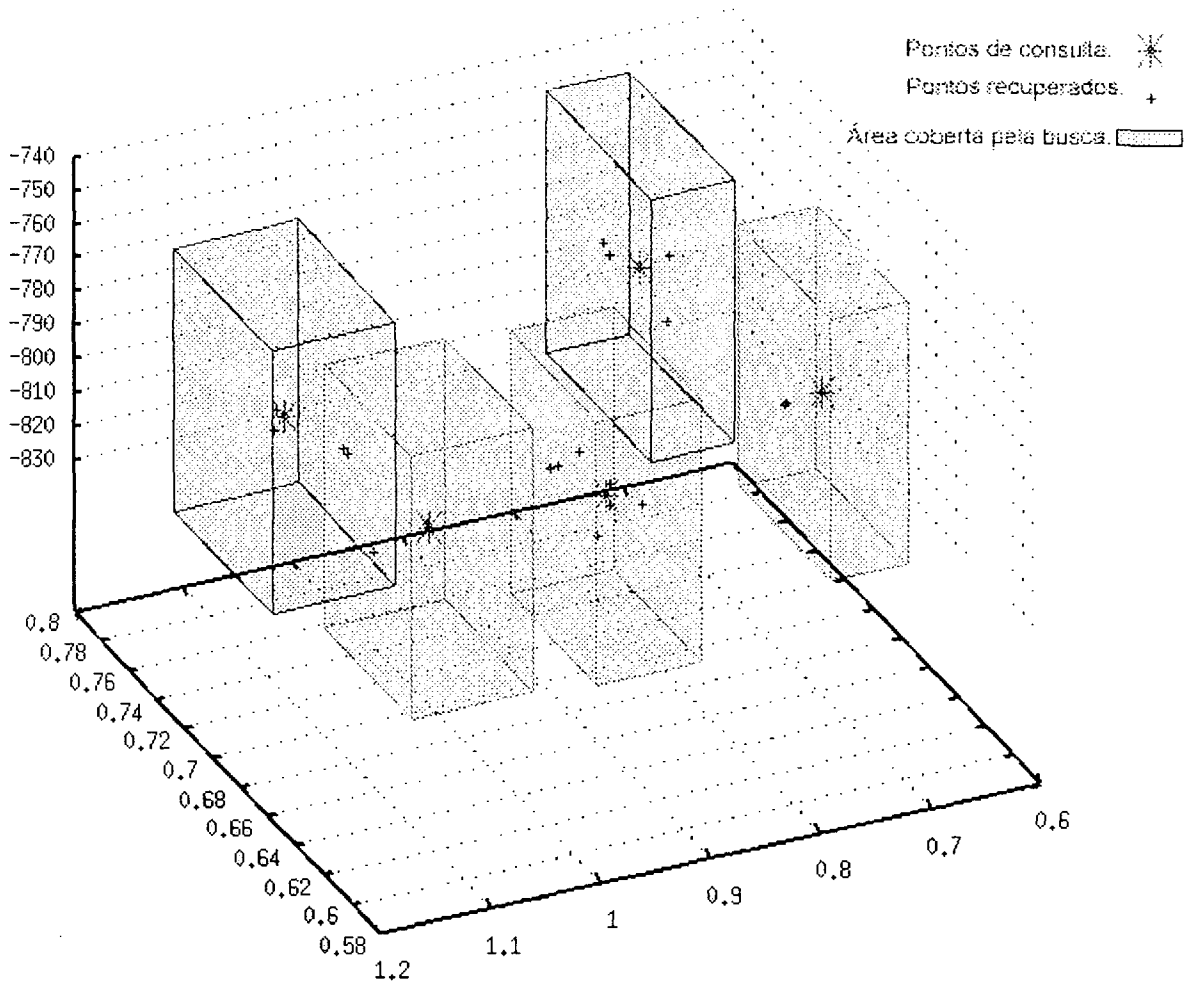


Gráfico 6b: Pesquisa por pontos similares usando a Estratégia 1.

O Gráfico 6c, apresenta a Estratégia 2, onde existe um percentual de tolerância para cada dimensão, permitindo assim realizar um ajuste melhor de acordo com a faixa de valores com a qual cada característica possua. Os valores estão registrados na Tabela 8, da Seção 6.7.

Pode-se observar uma melhor seleção nesta estratégia, pois é possível alterar as dimensões em diferentes proporções de acordo com a característica que está sendo avaliada.

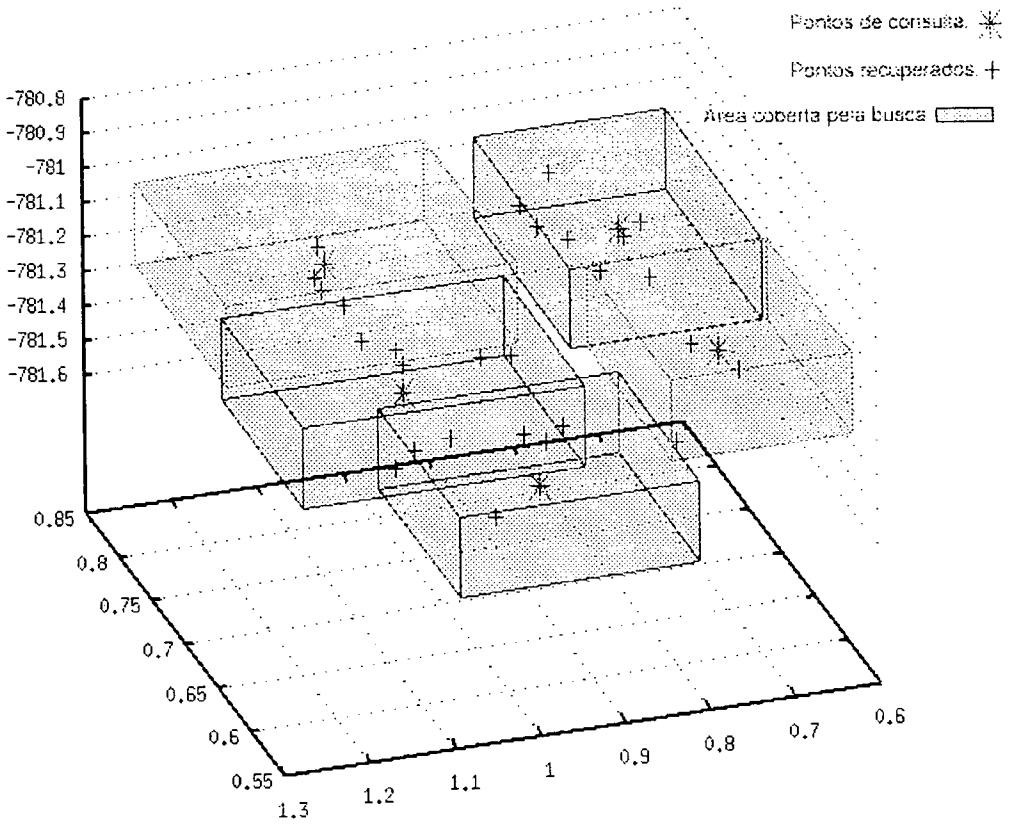


Gráfico 6c: Pesquisa por pontos similares usando a Estratégia 2.

O Gráfico 6d apresenta os dados referentes ao teste utilizando a Estratégia 3.

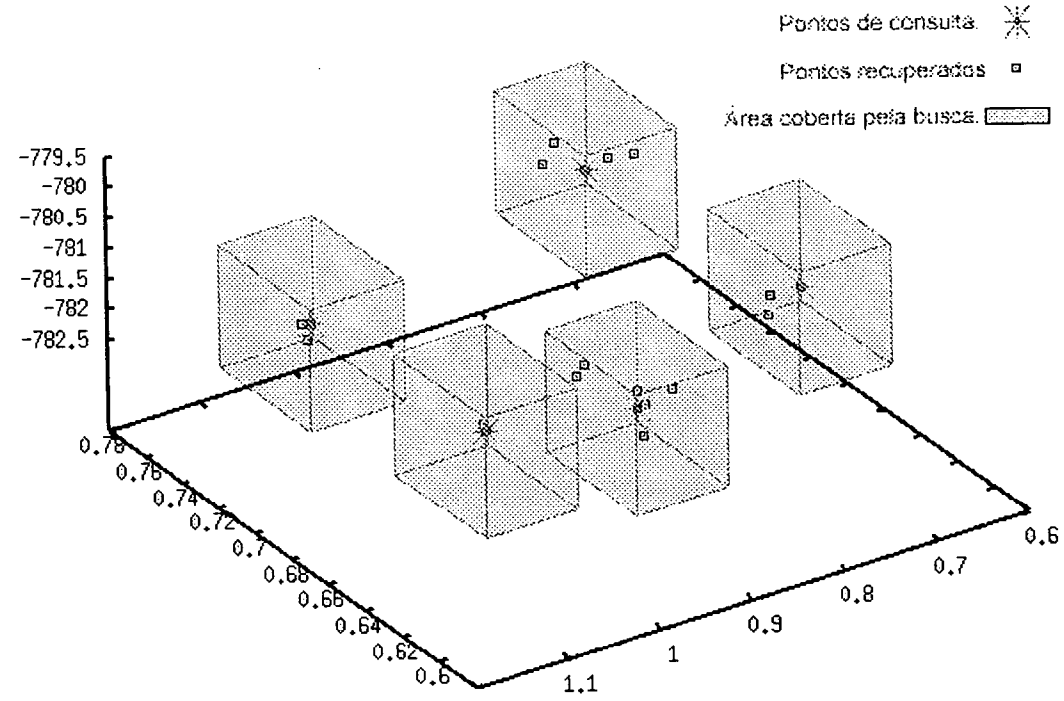


Gráfico 6d: Pesquisa por pontos similares usando a Estratégia 3.

Esta estratégia consiste em utilizar valores fixos para cada dimensão em vez de usar valores proporcionais ao valor procurado, refletindo desta forma os valores dos vizinhos mais próximos do ponto de consulta.

Ainda assim, é preciso otimizar as funções que determinam a similaridade pois, como são muito semelhantes às consultas por faixas de valores, em muitos casos, o índice criado não é utilizado.

7.7 – BUSCA POR IMAGEM EXEMPLO

Para completar o conjunto de testes, foram realizadas algumas consultas utilizando uma imagem como exemplo e observando as imagens resultantes da consulta. Neste caso, como ainda não foi identificada uma relação mais estreita entre as características extraídas, o resultado não reflete o desempenho do SGBD ou da estrutura criada e sim a necessidade de se obter uma função que determine a similaridade de imagens *range*.

Neste caso, foi utilizada a Estratégia 3 por parecer mais fiel à recuperação de imagens similares e permitir ajustes independentes por dimensão. Os resultados obtidos estão dispostos no Anexo IV.

8. RESULTADOS OBTIDOS E ESPERADOS

Os critérios usados para avaliar o desempenho das estruturas de indexação dizem respeito à quantidade de espaço utilizado para armazenar as informações da árvore, o número de acessos aos nós na busca por determinado objeto e a quantidade de recursos computacionais utilizados na manutenção da árvore (inclusões e exclusões).

A utilização de R-Trees com pontos é interessante pois minimiza o *espaço morto* (ou vazio) coberto por um nó, diminuindo o tempo de busca e também a sobreposição, diminuindo assim o número de ramos a serem percorridos durante uma pesquisa.

Conforme apresentado anteriormente, o que se espera de um método de acesso é um desempenho característico de uma estrutura convencional, como a "B-Tree". Neste trabalho os objetos criados, fundamentais no desempenho do método de acesso, mantiveram o resultado esperado, seja na recuperação dos dados, seja no atendimento às consultas formuladas.

Um problema comum dos SGBD é o cálculo utilizado pelo otimizador de consultas, pois na maioria dos casos, se existe dúvida entre usar o índice e fazer uma leitura seqüencial na tabela, o SGBD sempre opta pela segunda, pois ele sabe exatamente qual será o custo desta operação. E através deste teste ficou evidente que a função que determina o custo da pesquisa pelo método de acesso, deve ser revista para auxiliar melhor o otimizador. Com isso, pode-se afirmar que a imprecisão dos operadores criados, com relação à quantidade de operações que precisam ser realizadas para responder a uma consulta, pode dificultar a criação de uma função para determinar o custo SQL de uma consulta que utiliza este operador.

Nos testes realizados, o sistema foi submetido à realização de buscas por uma faixa de valores e observou-se que na maioria dos casos a busca se dá pela leitura total da tabela (*full scan*). Apenas quando a faixa de valores é estreita, é que o SGBD

utilizava o índice. Isso é comum também com as estruturas B-Tree. Porém é necessário aperfeiçoar a consulta por faixa de valores.

Na realização dos testes das estratégias surgiu a necessidade de um novo operador ou função capaz de recuperar informações onde um pequeno número de dimensões seja diferente. Por exemplo, caso o vetor de características manipulado possua dez dimensões e apenas uma delas seja um pouco divergente da estratégia utilizada, uma única dimensão divergente poderia eliminar uma boa candidata.

Outras funções poderiam auxiliar na identificação de relações entre as características das imagens *range*, tais como as funções de agregação *(máximo, mínimo, somatório, média)* ou funções que identifiquem a quantidade de dimensões divergentes entre dois pontos multidimensionais.

O resultado obtido foi bastante satisfatório em função das características que uma estrutura de indexação deve possuir. E a flexibilidade dos operadores, principalmente a criação de um novo tipo de operador para vetores de características, sendo totalmente configurável, permitiu a avaliação de estratégias.

9. CONCLUSÕES

Este trabalho apresenta um estudo amplo sobre os requisitos fundamentais, necessários para a implementação de um Sistema de Recuperação de Imagens baseada em Conteúdo (SRIC), destacando principalmente, a extração de características e os métodos de acesso às informações armazenadas. Estes elementos, conforme apresentado anteriormente, têm grande influência no desempenho do sistema. De maneira geral, foram apresentados alguns exemplos de sistemas existentes citando os ajustes a serem feitos para que um SRIC seja capaz de manipular imagens de profundidade.

Com este objetivo, foram discutidas as características comuns que podem ser extraídas de imagens de profundidade apresentando uma estrutura para o armazenamento destas informações. Por se tratar de uma estrutura complexa, concluiu-se que a forma mais eficiente para armazenar estas informações é em vetores de características, sendo necessária a criação deste tipo de dados em um banco de dados adequado ao armazenamento destas imagens.

Para possibilitar a recuperação das imagens, é necessário criar uma estrutura de índices que seja eficiente, permitindo que sejam encontradas todas as imagens que possuam as características similares às informações fornecidas como critério de busca. Para isso, a estrutura deve ser implementada sobre o tipo de dados da informação armazenada.

Desta forma, a linha mestra deste trabalho foi a avaliação de diversas estruturas de indexação que poderiam ser utilizadas na indexação das informações de imagens de profundidade, levando em consideração todos os aspectos de implementação e otimização de um banco de dados para um SRIC e estendendo a aplicação desta estrutura a um banco de imagens de profundidade.

Uma das principais preocupações deste trabalho foi a definição de um ambiente estável e flexível para avaliar as relações entre as características visuais e

espaciais extraídas deste tipo de imagem. Permitindo, desta forma, que o SRIC3D, desenvolvido pelo grupo IMAGO, tenha condições de avançar em suas pesquisas na reconstrução de modelos tridimensionais baseada em imagens de profundidade.

Para implementar tanto as estrutura de dados como as de indexação sobre os dados, foram avaliados alguns bancos de dados, procurando identificar qual deles seria capaz de prover a estrutura necessária para o armazenamento dos dados e que fosse bastante flexível para permitir a criação de novos tipos de dados e novas estruturas de indexação para estes novos tipos. Sem dúvida nenhuma, o POSTGRES foi a melhor opção. Por se tratar de um banco de dados objeto-relacional, provê todas as funcionalidades e características de um banco de dados relacional moderno, permitindo inclusive um controle transacional. Além disso, as características de orientação a objetos contidas no POSTGRES (polimorfismo, herança, etc.) permitiram a criação dos novos tipos de dados e de estruturas de indexação.

Outra vantagem do POSTGRES, identificada após minucioso estudo sobre a forma como as estruturas de dados são armazenadas, é a possibilidade de redefinir o comportamento das funções utilizadas para a implementação e armazenamento das estruturas de indexação. Sendo necessárias pequenas alterações nestas funções para a utilização satisfatória das estruturas de indexação existentes, com os tipos de dados definidos. Foi possível identificar, também, que o POSTGRES permite implementar qualquer tipo complexo de dados e construir uma estrutura de indexação sobre este tipo.

Com relação a estrutura implementada, pode-se observar que ela permite a inclusão de novas características facilitando o desenvolvimento de varias pesquisas em paralelo, sem que haja interferência de uma na outra, pois a estrutura criada se utiliza de tabelas temporárias e o isolamento entre seções permitindo que seções possam operar com estratégias diferentes. Esta estrutura permite ainda, com pequenas adaptações, que modelos gerados das imagens cadastradas no banco de dados podem

ser incluídos e usados para a recuperação das imagens de profundidade que compõem o modelo gerado ou são de um objeto similar.

No que se refere à estrutura de indexação para o tipo de dados vetor de características, a estrutura proposta pode ser facilmente configurada pelo módulo de mineração de dados (Figura 1), determinando os parâmetros a serem configurados nas estratégias de busca, ou permitindo a modificação das estratégias criadas, auxiliando a determinação de uma função de similaridade entre as características que as imagens de profundidade possuem e que não são comumente trabalhadas em bancos de dados convencionais.

Além da aplicação direta na recuperação de imagens de profundidade e, futuramente, seus modelos tridimensionais, a atual estrutura de dados e de indexação, pode ser utilizada em outros SRICs de aplicações diferentes, pois implementa um tipo de dado bastante comum na representação de imagens: o vetor de características. Esta mesma estrutura abre ainda inúmeras possibilidades de pesquisa, pois novos operadores precisam ser criados para realizar operações geométricas com estes dados, pois não existe ainda um conjunto padrão de operadores que atenda a todos os tipos de consultas possíveis de serem submetidas ao banco de dados.

10. ANEXOS

Anexo I : Tipos de dados do PostgreSQL exceto os tipos Data/Hora.

Categoria	Nome	Tipo	Tamanho
Numérico	int	Inteiro	2,4,ou 8 bytes
Numérico	float	Ponto flutuante	4 ou 8 bytes
Numérico	money	Moeda (currency)	4 bytes
Caractere	char	Caractere	1 byte
Caractere	char(n)	Caractere tamanho fixo	(4 + *n*) bytes
Caractere	text	Caractere tamanho variável	Sem limite
Caractere	varchar(n)	Caractere tamanho variável com limite	Máximo 104857690 bytes
Lógico	bool	Boolean *TRUE* ou *FALSE*	1 byte
Geométricos	point	Ponto no espaço	16 bytes
Geométricos	line	Linha infinita	32 bytes
Geométricos	lseg	Segmento de linha finito	32 bytes
Geométricos	box	Retângulo no espaço	32 bytes
Geométricos	path	Lista de pontos formando uma linha	(4 + 32n) bytes
Geométricos	polygon	Lista de pontos formando um polígono	(4 + 32n) bytes
Geométricos	circle	Centro do círculo e seu raio	24 bytes

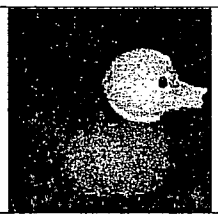
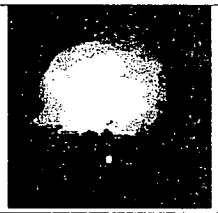
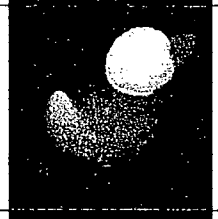
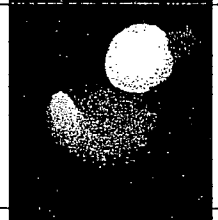
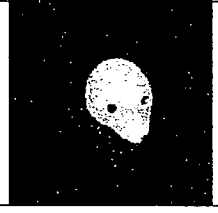
Anexo II: Operadores Geométricos do PostgreSQL

Operator	Description	Usage
+	Translação	box '((0,0),(1,1))' + point '(2.0,0)'
-	Translação	box '((0,0),(1,1))' - point '(2.0,0)'
*	Escala ou Rotação	box '((0,0),(1,1))' * point '(2.0,0)'
/	Escala ou Rotação	box '((0,0),(2,2))' / point '(2.0,0)'
#	Interseção	'((1,-1),(-1,1))' # '((1,1),(-1,-1))'
#	Número de Pontos em um polígono.	# '((1,0),(0,1),(-1,0))'
##	Pontos de maior proximidade	point '(0,0)' ## lseg '((2,0),(0,2))'
&&	Estão sobrepostos?	box '((0,0),(1,1))' && box '((0,0),(2,2))'
&<	Sobrepõe à esquerda?	box '((0,0),(1,1))' &< box '((0,0),(2,2))'
&>	Sobrepõe à direita?	box '((0,0),(3,3))' &> box '((0,0),(2,2))'
<->	Distancia entre	circle '((0,0),1)' <-> circle '((5,0),1)'
<<	Está à esquerda?	circle '((0,0),1)' << circle '((5,0),1)'
<^	Está à abaixo?	circle '((0,0),1)' <^ circle '((0,5),1)'
>>	Está à direita?	circle '((5,0),1)' >> circle '((0,0),1)'
>^	Está acima?	circle '((0,5),1)' >^ circle '((0,0),1)'
?#	Interseciona ou sobrepõe	lseg '((-1,0),(1,0))' ?# box '((-2,-2),(2,2))';
?-	Está horizontal em relação a ?	point '(1,0)' ?- point '(0,0)'
?-|	É perpendicular?	lseg '((0,0),(0,1))' ?-| lseg '((0,0),(1,0))'
@-@	Comprimento ou circunferência	@-@ path '((0,0),(1,0))'
?|	Está vertical em relação a ?	point '(0,1)' ?| point '(0,0)'
?||	São paralelas?	lseg '((-1,0),(1,0))' ?|| lseg '((-1,2),(1,2))'
@	Está contido ou dentro?	point '(1,1)' @ circle '((0,0),2)'
@@	Determina o centro do círculo	@@ circle '((0,0),10)'
~=	mesmo que	polygon '((0,0),(1,1))' ~= polygon '((1,1),(0,0))'

Anexo III: Operações suportadas pelos operadores implementados no Postgres

Operator	Description	Usage
&&	Estão sobrepostos?	bigbox_ops
&<	Sobrepõe à esquerda?	bigbox_ops
&>	Sobrepõe à direita?	bigbox_ops
<<	Está à esquerda?	bigbox_ops
>>	Está à direita?	bigbox_ops
@	Está contido ou dentro?	bigbox_ops
~=	Mesmo que	bigbox_ops
&&	Estão sobrepostos?	box_ops
&<	Sobrepõe à esquerda?	box_ops
&>	Sobrepõe à direita?	box_ops
<<	Está à esquerda?	box_ops
>>	Está à direita?	box_ops
@	Está contido ou dentro?	box_ops
~=	Mesmo que	box_ops
&&	Estão sobrepostos?	poly_ops
&<	Sobrepõe à esquerda?	poly_ops
&>	Sobrepõe à direita?	poly_ops
<<	Está à esquerda?	poly_ops
>>	Está à direita?	poly_ops
@	Está contido ou dentro?	poly_ops
~=	Mesmo que	poly_ops

Anexo IV: Resultado da Consulta por imagem exemplo.

duck-W+5-deg124.range (*Imagem Exemplo*)	
Imagens retornadas	
brain-w+5-set2-deg241.20.range.gif	
brain-W+5-set2-deg223.20.range	
duck-W+6-deg121.range	
duck-W+6-deg125.range	
duck-W+6-deg21.range	

11. REFERÊNCIAS BIBLIOGRÁFICAS

[1] ADORAM, M.; LEW, M. S.; "*IRUS: image retrieval using shape*", Proceedings of IEEE Multimedia Systems, 1999.

[2] AGGARWAL, C. C. "*On the Effects of Dimensionality Reduction on High Dimensional Similarity Search*", In ACM PODS Conference, 2001.

[3] AHMAD, I.; GROSKY, W. I.; "*Spatial Similarity-based Retrievals and Image Indexing By Hierarchical Decomposition*". In Proceedings of the International Database Engineering and Application Symposium (IDEAS'97), pp. 269-278, 1997.

[4] AKSOY, S.; HARALICK, R. M.; "*Using Texture in Image Similarity and Retrieval*", In Proceedings of International Workshop on Texture Analysis in Machine Vision, pp 111-117, 1999.

[5] AKSOY, S.; HARALICK, R. M.; "*Probabilistic vs. Geometric Similarity Measures for Image Retrieval*". In Symposium of Computer Vision and Pattern Recognition, pp 2357-2362, 2000.

[6] ALEXANDROV, D.; MA, W. Y.; EL ABBADIM, A.; MANJUNATH, B. S., "*Adaptive filtering and indexing for image databases*", In Proceedings of the SPIE International Conference on Storage and Retrieval for Image and Video Databases - III, pp 12--23, 1995.

[7] ALWIS, S. ; AUSTIN, J., "*An integrated framework for trademark image retrieval using gestalt features and CMM neural network* ". In 7th International Conference on Image Processing And Its Applications, Vol. 1, 1999.

[8] ARAÚJO, A. A.; GUIMARÃES, S.J.F. "*Recuperação de informação visual com base no conteúdo em imagens e vídeos digitais*". *Revista de Informática Teórica e Aplicada - RITA*, UFRGS, Porto Alegre–RS, Brasil, 2000.

[9] BAB-HADIASHAR,A.; SUTER, D. "*Data Segmentation and Model Selection for Computer Vision*", Springer-Verlag, New York, 2000.

[10] BECKMANN ,N.; KRIEGEL, H.P.; SCHNEIDER, R.; SEEGER , B. "*The R*-tree: an efficient and Robust Access Method for Points and Rectangles".* In Proceedings of ACM-SIGMOD International Conference on Management of Data, pp. 322-331, 1990.

[11] BELLON, O.R.P.; DIRENE, A.; SILVA, L., "*Edge detection to guide range image segmentation by clustering techniques*". In *Proc. 6th IEEE Int. Conf. on Image Processing*. v.2, p.725–729, Japão, 1999.

[12] BELLON, O.R.P.; SILVA, L.; GOTARDO, P. "*Edge-based segmentation using curvature sign maps of reflectance and range images*". In Proceedings of 8th IEEE International Conference on Image Processing – ICIP Thessaloniki-Greece. v.1, p.730 –734, Grécia, 2001.

[13] BERRETTI, S.; BIMBO, A. Del; PALA, P. . "Retrieval by Shape Using Multidimensional Indexing Structures". In Proceedings of 10th International Conference on Image Analysis and Processing (ICIAP'99), 1999.

[14] BRANDT, S. "Use shape features in content based image retrieval". M.Sc. Thesis, University of Helsinki, Department of Engineering Physics and Mathematics, 1999.

[15] BRES, S.; JOLION, J.M., "*Detection of interest points for image indexing*". In 3rd International Conference on Visual Information Systems, Visual 99, pp 427-434, 1999.

[16] BROWN, L.; GRUENWALD, L., "*Tree-Based Index for Image Data*", In Journal Of Visual Communication And Image Representation Vol. 9, No. 4, December, pp. 300–313, 1998.

[17] CASTLEMAN, K.R."Digital Image Processing", Prentice-Hall, 1996.

[18] CHANDRASEKARAN, S.; MANJUNATH, B. S.; WANG, Y. F.; WINKELER, J. e ZHANG, H., "*An eigenspace update algorithm for image analysis*". In Proceedings of the IEEE International Symposium on Computer Vision, pp 551-556, 1995.

[19] CIACCIA, P.; PATELLA, M.; RABITTI, F.; ZEZULA, P. "Indexing Metric Spaces with M-Tree". In Proc. Atti del Quinto Congresso Nazionalle SEBD, pp 67-86, June 1997.

[20] CURLESS, B.; LEVOY, M., "*A Volumetric Method for Building Complex Models from Range Images*", Proceedings of SICGRAPH 96, pp 303-312, 1996.

[21] DAGENHART, W. D.; KARR, K.; ROLLI, S. e SLIWA, K., "*A Solution for Data Handling Based on an Object Oriented Database System and its Applications to CDF RUN II*", In Stauts Report of Objectivity/DB Working Group. Technical Report CDF/DOC/COMP UPG/PUBLIC/4346, Fermi National Accelerator Laboratory, 1996.

[22] DATE, C. J., "*Introdução aos Sistemas de Bancos de Dados*",Rio de Janeiro: Ed. Campus, 1984.

[23] FALOUTSOS, C. , "*Searching multimedia databases by content*", Kluwer Academic, 1996.

[24] FALOUTSOS, C.; LIN, K-I., "*FastMap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets*", In ACM SIGMOD, pp. 163-174, 1995.

[25] FLICKNER, M.; SAWHNEY, H.; NIBLACK, W. ; ASHLEY, J. ; HUANG, W.; DOM, B.; GORKANI, M.; HAFINE, J.; LEE, D.; PETKOVIC, D.; STEELE, D.; YANKER, P.. "*Query by image and video content: the QBIC system*". In IEEE Computer, September, 1995.

[26] FISHER, R. B.; FITZGIBBON, A. W.; EGGERT, D., "*Extracting surface patches from complete range descriptions*". In Proceedings at the International Conference on Recent Advances in 3D Imaging and Modeling, 1997.

[27] FRAKES, W. B.; BAZEA-YATES, R., "*Information Retrieval – Data Structures & Algorithms*". Upper Sadlle River – New Jersey: Prentice-Hall , 1992.

[28] GAEDE, V.; GUNTHER, O., "*Multidimensional Access Methods*" In ACM Computing Surveys, 1998.

[29] GEVERS, T.; SMEULDERS, A.W.M., "*Image Indexing Using Composit Color and Shape Invariant Features*". In IEEE International Conference on Computer Vision (ICCV), pp 576--581, 1998.

[30] GONZALEZ, R. C.; WOODS, R. E., "*Digital Image Processing*", Addison Wesley, 1993.

[31] GUTTMAN, A., "*R-trees: A dynamic index structure for spatial searching*", In Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data, 1984, pp. 47–57.

[32] GÜNTHER, O.; BILMES, J.,"*Tree-based access methods for spatial databases: Implementation and Performance Evaluation*", IEEE Transac. Knowledge and Data Engineering. Vol. 3(3), pp:342-356,1991.

[33] HUANG, J.; KUMAR, S. R.; MITRA, M.; ZHU, W.-J.; ZABIH, R. , "*Image Indexing Using Color Correlograms*". In Proceedings IEEE Computer Society on Conference of Computer Visioin and Pattern Recognition, pp 762-768, 1997.

[34] IQBAL, Q.; AGGARWAL, J.K., "*Lower-Level and High-Level Approaches to Content-based Image Retrieval*". In Proceedings of SWS Image Analysis and Interpretation, pp 197-201, Abril, 2000.

[35] JAIN, A. K.,"*Fundamentals of Digital Image Processing*",Prentice-Hall, 1989.

[36] JEONG, S.; KIM, K.; CHUN, B.; LEE, J. ; BAE Y., J., "*An effective method for combining multiple features of image retrieval*". In TENCON 99 - *Proceedings of the IEEE Region 10 Conference*, Vol. 2, 1999.

[37] JIANG, X.; BUNKE, H., "*Fast Segmentation of Range Images into Planar Regions by Scan Line Grouping*". In Machine Visions and Applications, Vol. 7(2), pp 115—122, 1994.

[38] LIN, K-I; JAGADISH, H. V. e FALOUTSOS, C., "*The TV-tree: An index structure for high-dimensonal data*", In The Very Large Data Base Journal, vol. 3, pp: 517-542, 1994.

[39] KORNACKER, M.; SHAH, M.; HELLERSTEIN, J. M.. "*Research Track: An Analysis Framework for Access Methodds*". In Technical Report UCB/CSD-99-1051, University of California at Berkeley, 1999.

[40] LEUTENEGGER, S. T.; LOPEZ, M. A., "*The Effect of Buffering on the Performance of R-Trees*". In Proceedings of the 14th International Conference on Data Engineering, pp. 164--171, 1998.

[41] MA, W.; ZHANG, H., "*Benchmarking of Image Features for Content-based Retrieval*", In 32nd Asilomar Conference on Signals, Systems, and Computers, 1998.

[42] MALIK, J.; BELONGIE, S.; SHI, J. e LEUNG, T., "*Textons, contours and regions: Cue integration in image segmentation*". In Proceedings of International Conference on Computer Vision, pp. 918-925, 1999.

[43] MARSICOI, M. D.; CINQUE, L.; LEVIALDI, S., "*Indexing pictorial documents by their content: a survey of current techniques*". In Image and Vision Computing, Vol. 15, pp 119 – 141, February, 1997.

[44] MEHTRE, B.M.; KANKANHALLI, M.S.; LEE, W.F., "*Shape Measures for Content Based Image Retrieval: A Comparison*". In Information Processing and Management, Vol.33, No.3, pp. 319-337, 1997.

[45] MEHROTRA R.; GARY, J. E., "*Feature-Based Retrieval of Similar Shapes*", em Proceedings of 9th International Conference on Data Engineering (ICDE), pp. 108-115,1993.

[46] OGLE, V. E.; STONEBRAKER, M., "*Chabot: Retrieval from a Relational Databases of Images*". In IEEE Computer, Vol. 28, no. 09, pp. 40-48, 1995.

[47] PASS, G.; ZABIH, R.; MILLER, J., "*Comparing Images Using Color Coherence Vectors*". In ACM Conference on Multimedia, 1996.

[48] PENTLAND, A.; PICARD, R.; SCLAROFF, S., "*Photobook: Content-Besed Manipulation of Image Databases*". In SPIE Storage and Retrieval Manipulation of Image & Video Databases II, San Jose, CA, pages: 34-47, 1994.

[49] PEREIRA ,S. V.,"Avaliação de Sistemas de Recuperação de Imagens Baseada em Conteúdo: Um Estudo de Caso na Área Médica". Dissertação de Mestrado, Universidade Federal do Paraná - UFPR, 2001.

[50] PETRAKIS , E. G. M., "Fast Retrieval by Spatial Structure in Image Data Bases", Journal of Visual Languages and Computing (accepted, to appear),2002.

[51] PETRAKIS, E. G.M.; FALOUTSOS, C.; LIN, K-I., "*ImageMap: An Image Indexing Method Based on Spatial Similarity*". In IEEE Transactions on Knowledge and Data Engineering, Vol. 14, No. 5, pp. 979-987, 2002.

[52] RAMAMOORTHI, R.; ARVO, J., "*Creating generative models from range images*". In Proceedings of SIGGRAPH 99, pp 195-204, 1999.

[53] RUI , Y., T.S. HUANG,; CHANG, S-F., "*Image Retrieval: Current Techniquess, Promising Directions, and Open Issues*" em Journal of Visual Communication and Image Representation, vol 10, no 1, pp. 39-62, 1999.

[54] SAMET, H., "*The design and analysis of spatial data structures*", Addison Wesley, 1994.

[55] SELLIS ,T., ROUSSOPOULOS,N; FALOUTSOS, C., "*The R + -Tree: A Dynamic Index for Multi-dimensional Objects*", em Proc. of the VLDB Conf., Brighton, England, pp. 3 – 17, 1987.

[56] SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S., "*Database System Concepts*".- 3ª Edição. MacGraw–Hill, 1997.

[57] SMEULDERS, A.; WORRING, M.; SANTINI, S.; GUPTA, A.; JAIN, R., "*Content-based image retrieval at the end of the early years*". In IEEE Transactions on Pattern Analysis and Machine Intelligence, vol22, No. 12, 2000.

[58] SMITH, J. R.; CHANG, S., "Tools and techniques for color image retrieval.", IS & T/SPIE Proc. Vol.2670, Storage & Retrieval for Image and Video Databases IV, 1995.

[59] SMITH, J. R.; CHANG, S., "VisualSeek: A Fully Automated Content-Based Imge Query System", Proc. Of the ACM International Conference on Multimedia, 1996.

[60] SMITH, J. R.; CHANG, S., "Visually Searching the Web for Content", IEEE Multimedia, vol.4(3), 1997.

[61] SNYDER, J. M. *"Generative Modeling for Computer Graphics and CAD: Symbolic Shape Design Using Interval Analysis"*. San Diego: Academic Press, 1992.

[62] STAMOS, I.; ALLEN, P., "*3-D Model Construction Using Range and Image Data*", Proceedings of Computer Vision and Pattern Recognition, pp: 531—536, 2000.

[63] STONEBRAKER, M; ROWE, L. "*The design of Postgres*". In Proceedings of the SIGMOD Conference, June 1986.

[64] STRIKER, M.; ORENGO, M., "*Similarity of Color Images*". In Storage and Retrieval for Image and Video Databases (SPIE), pp. 381-392, 1995.

[65] SWETS, D. L.; PATHAK, Y.; WENG, J. J., "*A system for combining traditional alphanumeric queries with content-based queries by example in image databases*", In Technical Report. CPS-96-03, Michigan State University, 1996.

[66] THEODORIDIS, Y.; SELLIS, T., "*Optimization Issues in R-tree Construction*". In Proceedings of IGIS Symposium, 1994.

[67] TIAN, Q.; WU, Y.; HUANG, T. S., "*Combine User Defined Region-of-Interest and Spatial Layout for Image Retrieval*". In IEEE 2000 International Conference on Image Processing (ICIP'2000), pp. 746-749, Vol. 3, 2000.

[68] TRAINA JR, C.; TRAINA, A.; FALOUTSOS, C.; SEEGER, B., "*Fast Indexing and Visualization of Metric Datasets using Slim-Tree*". In IEEE Transactions on Knowledged and Data Engineering , Vol. 14, N.2, Abril 2002.

[69] TUCERYAN, M; JAIN, A. K., "*Texture Analysis*". In "*The Handbook of Pattern Recognition and Computer Vision (2nd Edition)*", by C. H. Chen, L. F. Pau, P. S. P. Wang (eds.), pp. 207-248 (Book Chapter), World Scientific Publishing Co., 1998.

[70] VIEIRA, E. V., "Mineração de Imagens em Profundidade", Dissertação de Mestrado, Departamento de Informática – Universidade Federal do Paraná, 2002.

[71] WEN, J.-R.; LI, Q.; MA, W.-Y.; ZHANG, H.-J.,"*A Multi-paradigm Querying Approach for a Generic Multimedia Database Management System*". In SIGMOD Record - Vol. 32(1): 26-34, 2003.

[72] WU, P.; MANJUNANTH, B.S.; NEWSAM, S.D.; SHIN, H.D. "*A texture descriptor for image retrieval and browsing*". In IEEE Workshop on Content-Based Access of Image and Video Libraries(CBAIVL '99), 1999.

[73] WHITE, D.; JAIN, R., "*Algorithms and strategies for similarity retrieval*". In Technical Report, Department of Computer Science and Engineering, University of California, San Diego, 1996a.

[74] WHITE, D.; JAIN, R., "*Similarity indexing: Algorithms and performance*", em Proceedings of SPIE Storage and Retrieval for Image and Video Databases, 1996b.

[75] YIANILOS, P., "*Data structures and algorithms for nearest neighbor search in general ametric spaces*". In Proceedings of the 4th ACM-SIAM Symposium on Discrete Algorithms, pp. 311-321, 1993.

[76] YU, Y.; FERENCZ, A.; MALIK, J., "*Extracting Objects from Range and Radiance Images*". In IEEE Transactions on Visualization and Computer Graphics, vol. 7, num. 4, pp 351-364, 2001.

[77] ZACHARY Jr, J.M.; IYENGAR, S.S.; "*Content Based Image Retrieval Systems*". In Application-Specific Systems and Software Engineering and Technology (ASSET), 1999.

[78] ZHANG, C.; CHEN, T., "*Active Learning for Information Retrieval:Using 3D Models As An Example*". In IEEE Transaction on Multimedia, Special Issue on Multimedia Database, April 2001.

[79] ZHOU, X.S.; COHEN, I.; TIAN, Q.; HUANG, S.T., "*Feature Extraction and Selection for Image Retrieval*". In ACM Multimedia, 2000.

[80] IMAGO – Grupo de Pesquisa em Processamento de Imagens e Visão Computacional do Departamento de Informática da Universidade Federal do Paraná – UFPR, em www.inf.ufpr.br/imago